

# Proactive Information Retrieval

Procheta Sen

B.Tech., M.Tech.

A dissertation submitted in fulfilment of the requirements for the award of

Doctor of Philosophy (Ph.D.)

to the



School of Computing  
Dublin City University

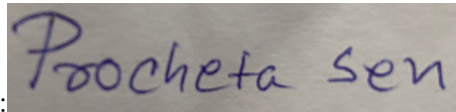
Supervisor: Prof. Gareth J.F. Jones

September 2021

## Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:



(Candidate) ID No.: 16213962

Date:10/09/2021

## Acknowledgments

I would like to thank my supervisor Professor Gareth Jones for his guidance throughout my PhD journey. I am also grateful to Debasis Ganguly who helped me to grow as a researcher during my PhD. I am also thankful to my external examiner Claudia Hauff for her insightful suggestions on my thesis.

I would also like to thank Mandar Mitra (ISI Kolkata) for his advice on my research work. I have learned a lot from all my collaborators specially Manisha Verma (Yahoo Research, New York), Dwaipayan Roy (IISER Kolkata), Killian Levacher (IBM Research Dublin), Yufang Hou (IBM Research Dublin) and Lea Deleris (Formerly IBM Research Dublin).

I am indebted to my parents for their love and sacrifice for my academic journey. Their unconditional support encouraged me to move forward during my PhD journey. I am also thankful to my friend Susan Robertson. Her love and support helped me to overcome the difficult times of Covid19 pandemic during my PhD. journey.

Finally I would like to acknowledge the following lines from the poem ‘The Road Not Taken’ written by the well know literary figure, Robert Frost, whose poems always inspired me during the trials and tribulations of my life Journey:

*Two roads diverged in a yellow wood,  
And sorry I could not travel both  
And be one traveler, long I stood  
And looked down one as far as I could*

...

*Two roads diverged in a wood, and I-  
I took the one less traveled by,  
And that has made all the difference.*

I am thankful to Science Foundation of Ireland for the research grant (grant 07/CE/I1142, Centre for Next Generation Localisation) to support this research.

# Contents

<b>Abstract</b>	<b>xii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Focus of this Thesis . . . . .	3
1.2 Scope of the Thesis . . . . .	5
1.3 Research Questions . . . . .	5
1.4 Structure of this Thesis . . . . .	7
<b>2 Background to Proactive Information Retrieval</b>	<b>9</b>
2.1 Information Retrieval . . . . .	9
2.1.1 Retrieval Models . . . . .	10
2.1.2 Relevance Feedback Models . . . . .	18
2.2 Proactive Information Retrieval . . . . .	24
2.2.1 Personal Information Management (PIM) . . . . .	24
2.2.2 Proactive Suggestion Methods . . . . .	26
2.2.3 Query Suggestion Methods . . . . .	29
2.3 Representation Learning for Text . . . . .	31
2.3.1 Language Modeling Techniquess for Word Semantics . . . . .	31
2.3.2 Word Embedding: A Generic Survey . . . . .	32
2.3.3 Word2vec Model Overview . . . . .	35
2.3.4 Contextual Word Embedding . . . . .	36
2.4 Summary . . . . .	38

<b>3</b>	<b>A Framework for Proactive Information Retrieval</b>	<b>39</b>
3.1	Proactive IR Generic Framework . . . . .	39
3.1.1	The Generic Workflow of a Proactive IR Model . . . . .	41
3.1.2	Information Sources for Proactive IR Model . . . . .	42
3.1.3	User's Personal History . . . . .	42
3.1.4	Similar Activity of Other Users . . . . .	43
3.2	Anatomy of a Proactive IR Model . . . . .	44
3.2.1	Proactive Query Formulation Model . . . . .	45
3.2.2	Generalization beyond Term Matching . . . . .	48
3.3	Concluding Remarks . . . . .	49
<b>4</b>	<b>Evaluation of Proactive Information Retrieval</b>	<b>50</b>
4.1	Evaluation Metrics in Traditional IR . . . . .	50
4.1.1	Set-based Evaluation Metrics . . . . .	51
4.1.2	Rank-based IR Measures Using the Concepts of Precision and Recall . . . . .	53
4.1.3	Evaluation Measure of Non-Binary Relevance Assessments . .	55
4.2	Differences Between Standard IR and Proactive IR Evaluation Criteria	56
4.3	Desirable Characteristics of Proactive IR Evaluation Metric . . . . .	57
4.4	Proposed Evaluation Metric . . . . .	59
4.4.1	Evaluating Proactive Queries . . . . .	60
4.4.2	Evaluating the Ranked List of Potentially Useful information Sources . . . . .	61
4.4.3	Analysis for Variations in $\pi$ -Index . . . . .	64
4.5	Reference Set ( $R_k$ ) Computation . . . . .	65
4.5.1	Reference Set Computation for Single Stage Task . . . . .	66
4.5.2	Reference Set Computation for Multi-Stage Task . . . . .	67
4.6	Concluding Remarks . . . . .	68

<b>5</b>	<b>Identifying Similar User Activities</b>	<b>69</b>
5.1	Introduction . . . . .	69
5.2	Task Extraction and Query Embedding . . . . .	72
5.2.1	Unsupervised Methods . . . . .	72
5.2.2	Supervised Methods . . . . .	74
5.3	Embedding Terms from User Activities . . . . .	75
5.3.1	Problems with Short Documents . . . . .	75
5.3.2	Word Vector Transformation with Semantic Contexts . . . . .	76
5.3.3	Constructing the Set of Task-specific Context . . . . .	78
5.3.4	Edge Weight Computation . . . . .	81
5.4	Clustering of Embedded Query Vectors . . . . .	83
5.4.1	Query Vector Embedding . . . . .	83
5.4.2	Clustering of Query Vectors . . . . .	84
5.5	Experimental Setup . . . . .	86
5.5.1	Dataset . . . . .	86
5.5.2	Baselines and Experiment Objectives . . . . .	89
5.5.3	Parameters and Evaluation Metrics . . . . .	92
5.6	Experimental Results . . . . .	94
5.6.1	Within-session task extraction . . . . .	96
5.6.2	Cross-Session Task Extraction . . . . .	98
5.7	Concluding Remarks . . . . .	100
<b>6</b>	<b>Proactive Suggestion For Single Stage Tasks</b>	<b>101</b>
6.1	Background and Motivation . . . . .	102
6.1.1	Simulation Setup Used in IR . . . . .	103
6.1.2	Associative Document Retrieval . . . . .	104
6.2	Workflow of Proactive Agent for Single Stage Tasks . . . . .	105
6.2.1	Interaction Environment . . . . .	105
6.2.2	Active Set . . . . .	106

6.2.3	User Actions . . . . .	107
6.2.4	Proactive Recommendation Agent . . . . .	108
6.3	Simulation Setup With Document Relevance . . . . .	109
6.3.1	Simulated Reading Cycles . . . . .	111
6.3.2	Simulated Reading and Writing Cycles . . . . .	114
6.4	Query Formulation for Single Stage Task Setup . . . . .	116
6.5	Experimental Setup . . . . .	118
6.5.1	Experiment Objectives . . . . .	118
6.5.2	Dataset . . . . .	119
6.5.3	Evaluation Setup . . . . .	121
6.5.4	Investigation of different Term Selection Approaches . . . . .	123
6.5.5	Query Formulation Components Analysis . . . . .	127
6.5.6	Implementation Details . . . . .	129
6.6	Results of Simulated Proactive IR . . . . .	130
6.6.1	QFM Performance Compared to Baselines . . . . .	131
6.6.2	Ablation Results on Simulated Read and Read-Write Activities	133
6.6.3	Parameter Sensitivity Analysis . . . . .	139
6.7	Conclusions . . . . .	142
<b>7</b>	<b>Proactive Suggestions For Multi-Stage Tasks</b>	<b>145</b>
7.1	Proactive Suggestion in Search Sessions . . . . .	147
7.1.1	Overall Architecture . . . . .	149
7.2	Proactive Query Formulation . . . . .	150
7.2.1	Graph Representation learning . . . . .	151
7.3	Experimental Setup . . . . .	154
7.3.1	Dataset . . . . .	155
7.3.2	Evaluation Outline . . . . .	158
7.3.3	Approaches Investigated . . . . .	159
7.3.4	Parameter Setting . . . . .	166

7.4	Results and Discussion . . . . .	168
7.4.1	Comparisons Between the Investigated Approaches . . . . .	169
7.4.2	Further Analysis . . . . .	172
7.5	Concluding Remarks . . . . .	177
<b>8</b>	<b>Conclusions</b>	<b>178</b>
8.1	Research Questions Revisited . . . . .	178
8.1.1	Proactive IR Framework . . . . .	178
8.1.2	Evaluating Proactive IR . . . . .	179
8.1.3	Grouping Similar activities of Users . . . . .	179
8.1.4	Proactive Support for Single Stage Tasks . . . . .	179
8.1.5	Proactive support in Multi-stage Task Scenario . . . . .	180
8.2	Future Work . . . . .	180
8.2.1	Chapter 3 . . . . .	181
8.2.2	Chapter 4 . . . . .	181
8.2.3	Chapter 5 . . . . .	181
8.2.4	Chapter 6 . . . . .	182
8.2.5	Chapter 7 . . . . .	182
8.3	Closing Remarks . . . . .	183
	<b>Appendices</b>	<b>184</b>
<b>A</b>	<b>Publications</b>	<b>185</b>
A.1	My Publications . . . . .	185
<b>B</b>	<b>Further Analysis of Proactive IR Evaluation Metric</b>	<b>187</b>
B.1	Introduction . . . . .	187
B.1.1	Sample Proactive Search Systems (PSS) . . . . .	187
B.1.2	Relevance Model based PSS (RM-PSS) . . . . .	187
B.1.3	Query Prediction based PSS (QP-PSS) . . . . .	188
B.1.4	Experiment Settings and Results . . . . .	188



<b>C</b>	<b>Evaluation of Proposed Word Embedding in Standard Task</b>	<b>192</b>
C.1	Experimental Setup . . . . .	192
C.1.1	Dataset . . . . .	192
C.1.2	Baselines . . . . .	193
C.2	Implementation Details . . . . .	193
C.3	Evaluation Tasks and Datasets . . . . .	195
C.3.1	Word Similarity. . . . .	195
C.3.2	Word Analogy. . . . .	196
C.3.3	Concept Categorization Task. . . . .	196
C.3.4	Evaluation Metrics and Pipeline. . . . .	197
C.4	Results . . . . .	198
C.5	Conclusions . . . . .	200

# List of Figures

1.1	Schematic overview of a proactive search system. . . . .	5
2.1	Schematic diagram of an IR System . . . . .	10
2.2	Example of the use of sliding window in computation of word embed- ding using word2vec . . . . .	35
2.3	Neural Architecture for Skipgram Word2Vec . . . . .	37
3.1	Sample Activity Log of a User . . . . .	40
3.2	Generic Framework for proactive IR. . . . .	41
3.3	Generative model for proactive Query Formulation. . . . .	46
5.1	Graph Constructed From a Pair of Consecutive Queries. . . . .	79
5.2	Sample queries from search sessions from the AOL query log. . . . .	86
5.3	Sample task labels from the AOL query log. . . . .	87
5.4	Sensitivity of context size with negative sampling for our proposed temporal-lexical word embedding. . . . .	88
5.5	Sensitivity of task clustering with variations in $\alpha$ (top) and $\eta$ (bottom). . . . .	95
6.1	Work Flow of Proactive IR model in a Real User Setup. . . . .	109
6.2	Work Flow of Proactive IR Model in Our Proposed Simulated User Setup. . . . .	115
6.3	Sample topic and manually judged relevant sentences for the corre- sponding topic 305 in TREC Novelty Track Dataset. . . . .	121
6.4	Variations in P@5 for different values of $k$ and $b$ in RLM . . . . .	127

6.5	Effect of varying the number of query terms in P@5 at step 1 in simulated read activities. . . . .	128
6.6	Effect of varying the word embedding dimension in QFM and KDERLM for simulated read activities in step 1. . . . .	129
6.7	Effect of varying the number of expansion terms in P@5 at step 1 in simulated read activities. . . . .	130
6.8	P@5 at different steps of proactive suggestions in ‘R’ and ‘R/W’ activities . . . . .	132
6.9	MRR at different steps of proactive suggestions in ‘R’ and ‘R/W’ activities . . . . .	133
6.10	Cumulative Recall at different steps of proactive suggestions in ‘R’ and ‘R/W’ activities . . . . .	134
6.11	Variations in the values of MRR, P@5 and Cumulative Recall with time progression (i.e. increase in steps) in simulated setup. . . . .	135
6.12	Variations in MRR, in Different Steps for QFM, QFM+Okapi, QFM+RLM and QFM+KDERLM . . . . .	138
6.13	Variations in P@5 in Different Steps for QFM, QFM+Okapi, QFM+RLM and QFM+KDERLM . . . . .	138
6.14	Variations in Cumulative Recall in Different Steps for QFM, QFM+Okapi, QFM+RLM and QFM+KDERLM . . . . .	139
6.15	Effect of varying the k in P@5(Top Diagram) and MRR (Bottom Diagram) values for simulated write cycles in Step 1. . . . .	140
6.16	Effect of varying the k in P@5(Top Diagram) and MRR (Bottom Diagram) values for simulated read write cycles in Step 1 . . . . .	141
6.17	Effect of applying true relevance feedback and true relevance feedback along with pseudo relevance feedback in the 2 <sup>nd</sup> pass of simulated read activity for QFM . . . . .	142
6.18	Effect of varying the mixing parameter $\alpha$ on P@5 in simulated read activities at step 1. . . . .	142

6.19	Effect of varying the mixing parameter $\alpha$ on MRR in simulated read activities at step 1. . . . .	143
6.20	Effect of varying the mixing parameter $\alpha$ on cumulative recall in simulated read activities at step 1. . . . .	143
7.1	Sample queries from AOL query log. . . . .	149
7.2	Schematic Diagram of a specific proactive search system (PSS). . . .	149
7.3	A sample sub-graph constructed from the AOL queries of Figure 7.1 .	153
7.4	Sensitivity of proactive IR effectiveness with variations in the number queries considered to consider set $H_t$ . . . . .	169
7.5	Sensitivity of proactive IR effectiveness with variations in $\pi$ , i.e., relative proactivity duration. . . . .	172
7.6	Sensitivity of proactive IR effectiveness with variations in $\beta$ , i.e., importance of user's own past queries compared to similar queries from other users. . . . .	173
7.7	Sensitivity of proactive IR effectiveness with variations in the number of terms in proactive query. . . . .	174
7.8	AP, $\rho$ and MRR values of GRLW for different ranges of similarities between consecutive queries. . . . .	175
B.1	Comparison of PREVAL-RR values of RM-PSS (x-axis) and QP-PSS-W (y-axis), $\pi$ -index values being shown alongside the axes. . . . .	190

# List of Tables

3.1	Notations For Describing Proactive IR Model . . . . .	44
4.1	Notation for proactive evaluation metric. . . . .	61
5.1	Dataset statistics of task annotated queries from the AOL query log.	90
5.2	Comparison between the best results obtained after parameter tuning on different unsupervised approaches of within session task extraction. *†‡ indicates statistical significance (paired t-test with 95% confidence) with respect to $QC_{WCC}$ using Wikipedia, ‘Qry-vec temporal context’ and ‘Qry-vec tempo-lexical context’ respectively. . . .	96
5.3	Comparison between different approaches for cross-session task extraction . . . . .	96
5.4	Nearest Neighbors for the word ‘jfk’ . . . . .	98
6.1	Types and examples of user actions that change the active set of an environment. . . . .	108
6.2	Details of the TREC-2002 Novelty Track dataset used in our experiments on proactive IR. . . . .	119
6.3	Comparison of our proposed query formulation approach (i.e. QFM) with existing term selection approaches for simulated read and read-write activities. . . . .	131
6.4	Results of proactive IR effectiveness with simulated read activities of user agents. . . . .	136

6.5	Results of proactive IR effectiveness with simulated read/write activities of user agents. . . . .	137
7.1	Training and Test splits of the AOL Query log . . . . .	158
7.2	Summary of the approaches investigated. . . . .	166
7.3	Comparisons between the query prediction and document retrieval effectiveness obtained with different query prediction and proactive document retrieval methods. . . . .	168
7.4	Investigating BERT-based passage reranking on retrieval effectiveness obtained with our proposed approaches. . . . .	176
B.1	PREVAL metrics computed over the evaluation set. . . . .	189
C.1	Dataset characteristics of DBPedia-2014. . . . .	193
C.2	Word analogy datasets overview. . . . .	196
C.3	Word similarity prediction results. . . . .	200
C.4	Word analogy results. . . . .	200
C.5	Concept categorization results. . . . .	201

# Proactive Information Retrieval

Procheta Sen

## Abstract

Users interact with digital systems with some task in his mind. An example of a task could be writing a research paper on a topic. Tasks can be single or multi-staged. In the process of accomplishing their task objectives, a user often needs to interact with an information retrieval (IR) system to address one or more information needs which arise while working on their task, e.g. for writing their research paper on a chosen topic, the user needs to look for existing research works related to the topic. Traditional IR systems do not take into account a user's task intent while showing search results to the user for a specific query submitted by the user. In our work we propose next generation IR systems (i.e. proactive IR systems) which seek to anticipate the user's underlying task from his interaction with a digital system to automatically identify their information needs and to suggest potentially relevant information sources to help the user to accomplish his task.

# Chapter 1

## Introduction

Users typically interact with a digital system with the goal of accomplishing a task. ‘Task’ can be defined as the overall information goal of the user. A task can either be single-stage or multi-stage depending on the number of steps required to accomplish it. An example of a single-stage task might be fixing the compilation error of a computer program, whereas an example of a multi-stage task could be planning for a vacation, which often involves a number of single-stage tasks such as booking flights, searching for places to visit, booking accommodation. In the process of accomplishing task objectives, a user often needs to interact with an information retrieval (IR) system to address one or more information needs which could arise while working on their task, e.g. for planning a vacation, a user may need to look for places to visit during their vacation, or searching for accommodation. Similarly, for a single-stage task, such as fixing a compilation error, a user may need to search for the possible causes of the error in an online forum.

Generally speaking, in a traditional IR or search system setup, a user is required to explicitly specify his information need in the form of a search query. Based on the query, an IR system typically returns a ranked list of documents which best match the query according to the algorithms used in the search system. The user then scrolls through the ranked list seeking content which satisfies their information need.



The retrieval results shown by a traditional IR system depend on the query formulation expertise of the user. A non-expert user may not get potential relevant information sources for his query because of poor query formulation strategy (White and Morris, 2007). Existing literature has explained approaches such as query expansion (Qiu and Frei, 1993) and search result re-ranking (Lavrenko and Croft, 2001) to enhance the query representation of the user. The assumption in search result re-ranking is that the top  $k$  documents retrieved for an initial query are relevant to that query, or more strictly the user’s information need. The performance of both query expansion and search result re-ranking depends on the quality of the initial query.

Another disadvantage of traditional IR systems is that they do not take the user’s task intent into account while determining search results to show to the user. The user’s task plays an important role in determining their information need (Feild and Allan, 2013). For example if a user searches for ‘python’ while writing code, then it is more likely that he is searching for items relating to the python programming language rather than the ‘snake python’.

A user’s writing code task will provide clues about their actual query intent in this scenario. In the web search literature, there is work (Feild and Allan, 2013; Ben Carterette et al., 2014; Van Gysel et al., 2016) which exploits the previous few queries typed by a user within a search session to enhance their current query. Since the focus of the work in (Feild and Allan, 2013; Ben Carterette et al., 2014; Van Gysel et al., 2016) is on improving the quality of the ranked list of retrieved documents, in terms of relevance to the current information need, there is no scope for estimating the overall information goal or task of the user. Knowing something of the overall information goal of the user may help in predicting what the user may need or find useful in their future activities.

Another thread of work (Yang et al., 2016; Shokouhi and Guo, 2015) has focused on providing search results in zero query scenarios. In this line of work, information snippets or cards are suggested to users based on their current context, e.g.,

activity, location. In one scenario the user’s overall topic of interest was displayed using information cards. These information snippets were found to be useful in this scenario. However, they could not support the user in accomplishing their current information goal.

To alleviate the limitations discussed above, this PhD focuses on next generation *proactive IR* (PIR). The concept of PIR was first described in (Rhodes and Maes, 2000). This study proposed *just in time information retrieval agents* which provide proactive suggestions to the user based only on his current context or activities. In our research scope a PIR model seeks to integrate search as an assistive component within a user’s task (both single and multi-staged), monitoring the user’s activities, and seeking to anticipate and address information needs in an automatic or semi-automatic way. An ideal PIR system would anticipate a user’s task intent and suggest potentially useful information related to his task (e.g. anticipate information related to travel planning or related programming resources) without the user being required to explicitly search for such information. In our research scope, understanding the user’s task intent refers to discovering the semantic concepts or topics that emerge by analyzing the user’s recent and current desktop activities. Proactive systems can either support users when they trigger a search process, where they realize that they have an information need, or entirely autonomously initiate a search and supply the user with potentially useful retrieved information that the user was unaware of or did not think to look for.

## 1.1 Focus of this Thesis

In our investigation, the notion of proactive support for single-stage tasks corresponds to that of anticipating a user’s information needs and automatically generating queries from the user’s recent desktop activities, such as informative content extracted from recently accessed files, etc. (in our example the source code file being compiled), and then executing these queries (e.g. terms extracted from the

erroneous regions of the source code) on a standard search system to retrieve relevant information (e.g. Stackoverflow<sup>1</sup> pages relevant to fixing the error). Note that the difference between this proactive approach and a standard search system is that the user does not need to explicitly enter a search query (i.e. in our example, the user may not need to copy-paste parts of the source code with terms that may help in retrieving relevant resources for the fix).

For multi-stage tasks, we seek to suggest useful information for a number of single stage tasks that are expected to be performed in conjunction with the current activity. A concrete example of this is a case when a user plans for a vacation and searches for flights booking along with booking accommodation, and looking for places to visit. Information about what other users have searched for when exploring the same topic as the current user could also support search for this user.

Let us now have a closer look at the pieces of information that may be useful in developing proactive methodologies for both single-stage and multi-stage tasks. The information sources that could potentially be useful can broadly be categorized into two different types:

1. Information from the *current user*. This refers to the desktop activities of the current user. To be more precise, logs of the user’s desktop activities, e.g. content he has been reading, queries entered, content he has been writing, may be indicators of potential information needs relating to their current activities.
2. Previous activity patterns related to *similar tasks from other users*. The second source of potentially useful information is activities of other users performed in past to accomplish similar objectives to those of the current user (Hassan Awadallah et al., 2014). This information source may be useful to provide proactive suggestions to the current user.

---

<sup>1</sup><https://stackoverflow.com/>

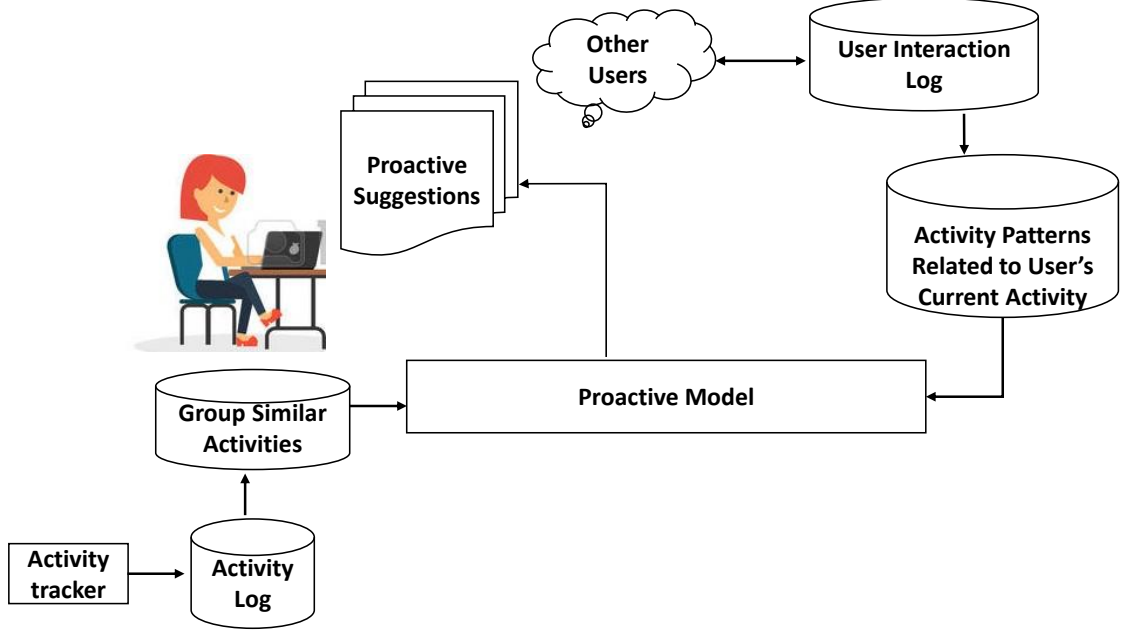


Figure 1.1: Schematic overview of a proactive search system.

## 1.2 Scope of the Thesis

In this section, we discuss how we plan to build our proactive model by exploiting the two different information sources described in Section 1.1. Figure 1.1 illustrates a schematic overview of a proactive IR model. It is clearly visible from Figure 1.1 that the proactive IR model potentially exploits two different information sources, the user’s current and recent activities, and similar activity patterns of other users and automatically formulates a user’s information need into a query on his behalf. The query anticipated by the proactive model is used to provide the user with a ranked list of suggestions. For a single stage task, the objective of proactive suggestion is to support a user with the topic in which he is currently engaged, and for multi stage tasks the objective is to provide support by fetching information related to different sub-tasks that may arise in the current multi-stage task of the user.

## 1.3 Research Questions

In this section, we develop the research questions which we will seek to address in our investigation of proactive IR models in this thesis. Evaluation is an essen-

tial component of research investigating new IR methods. An effective evaluation methodology will help us to investigate and build better proactive models. When using standard IR evaluation metrics (e.g. average precision, NDCG), we have a set of clearly defined user queries for which we evaluate the IR model based on its performance for these queries. However, in proactive IR, we may not have a query expressed by the user. In PIR, we try to anticipate the information need of a user in order to support them in their overall task. In this setting, all the user activities will be related to a similar information goal, and the PIR process may involve multiple queries related to the same user task. As a result of this, we cannot directly use standard IR evaluation metrics to evaluate a proactive IR model. Hence our first research question is as follows.

- **RQ1:** How can we evaluate the effectiveness of proactive IR models?

The rest of our research questions relate to the definition and exploration of the potential for development of effective proactive IR models. The first step of a proactive IR model is to estimate the current task which the user is seeking to accomplish. To do this, a first step involves identifying the activities (i.e. a user's interaction with a computing device like desktop or laptop) which relate to the user's current task. Hence we need to group the user's personal desktop activities, that are related to the same task. For example, activities such as writing a file document while reading another document may relate to the same task. We might also seek to identify activities of other users that are semantically related to the current user's recent and current activities. Relevant pieces of information extracted from these similar activity patterns of other users might be leveraged as information of use to the current user. Consequently, we state our second research question as follows:

- **RQ2:** Can we automatically group different desktop activities (e.g. file reading, file writing, user typed query etc.) that relate to the same underlying task or information goal?

Following identification of activities related to the user's overall task a PIR model

needs to proactively formulate a query to provide proactive suggestions to the user. Hence our third research question is:

- **RQ3:** Can we formulate the current information need of a user as a query using information extracted from their recent desktop activities, and use this to retrieve content relevant to the user, without them explicitly entering a query to an IR system?

We already described in Section 1.1 that other users' past activities which are similar to the current user's present activities, can also be used to provide proactive suggestions to the current user. Hence in our fourth research question, we investigate potential exploitation of the activities of other users in improving proactive suggestions for the current user. Our fourth research question is thus:

- **RQ4:** Can we provide useful information (e.g. documents, text snippets) to the current user by leveraging past activities of other users that are similar to the current user's activities?

## 1.4 Structure of this Thesis

The structure of this thesis is as follows:

- **Chapter 2:** Here we first revisit existing state-of-the-art literature on information retrieval. Then we describe existing work on proactive information retrieval. Since we apply word embedding in different approaches across all the chapters, we also revisit existing word embedding literature in this chapter.
- **Chapter 3:** There is no standard proactive IR framework in existing IR literature. In this chapter we describe the proactive IR framework used for our experiments. We introduce the assumptions in our proposed framework. Given the proactive IR framework we propose a generic proactive query formulation model.

- **Chapter 4:** In this Chapter, we addressed RQ1 which is ‘How can we evaluate a proactive IR model?’ We proposed a generic evaluation metric for the evaluation of proactive IR. Then we described how it can be applied in the two different scenario discussed in Chapter 1 : i) single stage task scenario and ii) multi-stage task scenario.
- **Chapter 5:** In this chapter we address RQ2 which is how can we bring user activities related to same information goal together. We propose a graph-based embedding framework to bring similar user activities together in this chapter.
- **Chapter 6:** In this chapter we addressed RQ3 which is how can we proactively support a user given his activity context. More precisely, we examine the effectiveness of our proactive query formulation model proposed in Chapter 3 in a simulated single stage task scenario. In single stage framework, a user is involved in a single information need.
- **Chapter 7:** In this chapter we address RQ4, which is how we can provide proactive suggestion using similar activities from other users. For multi-stage tasks, our assumption is that a user has multiple information needs in the task. More specifically, we investigate the effectiveness of our proactive query formulation model proposed in Chapter 3 in a web search setup.
- **Chapter 8:** Chapter 8 is the conclusion chapter. In this chapter we described the summary of each of the previous chapters (i.e. 4, 5, 6, 7). Then we describe the possible future directions from the work described in each chapter .

# Chapter 2

## Background to Proactive Information Retrieval

This chapter reviews the background of existing work relevant to the investigations described in this thesis. In Section 2.1, we give an overview of the state-of-the-art in information retrieval (IR) models. In Section 2.2, we motivate the use of proactive IR and subsequently describe existing work on proactive IR. In Section 2.3, we introduce the concept of embedding to provide semantic representation of words or sentences, which is used extensively in Chapters 4, 5 and 6 to compute semantic similarity between words in the methods examined in those chapters.

### 2.1 Information Retrieval

The subject of (IR) is concerned with the science behind the process of identifying potentially useful information sources to address the information need of a human user. In traditional IR setups, a user expresses their information need in the form of a search query. IR models seek to use this query to identify information sources potentially able to satisfy the user’s information need. Information sources can cover a wide range of modalities (e.g., text, images etc.), types (e.g. news articles, patents, research articles etc.), scope and coverage (e.g., long documents, short passages,



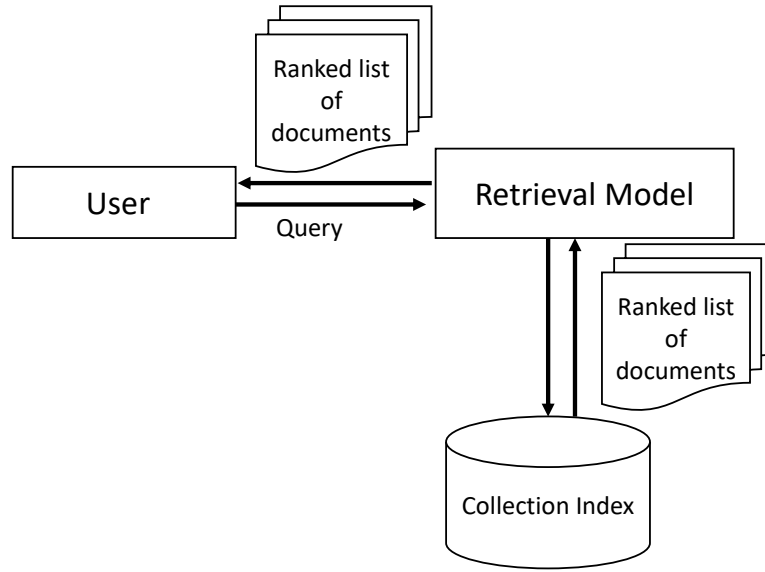


Figure 2.1: Schematic diagram of an IR System

single sentences etc.). The retrieval unit of an IR system depends on the type of information source used. Thus, a collection of information sources can be indexed at the level of documents, passages or sentences. Figure 2.1 depicts a schematic diagram of a typical IR system. Figure 2.1 illustrates how a retrieval model is used to compute the similarity between a query and a document in the collection index which is used as an estimate of the relevance of each document for a query. In the following subsection we introduce the principles adopted in standard IR models and their use in popular IR models which have been empirically demonstrated to be the most effective and hence the most commonly used ones.

### 2.1.1 Retrieval Models

The existing IR models described in the literature use different components (e.g. query term matching, document length, query term importance) to estimate the similarity between a query and a document. We introduce these models and their components in the following subsections.

## Boolean Retrieval Model

The oldest and the simplest retrieval model used in IR is the Boolean model (Manning et al., 2008). The query in a Boolean retrieval model is represented as a sequence of terms separated by Boolean operators, such as AND, OR and NOT. The retrieved documents, in this case, are those which satisfy the Boolean predicate function expressed by the query. For example, if the query is ‘relativity’ AND ‘theory’, the Boolean retrieval model retrieves documents containing both the terms ‘relativity’ and ‘theory’.

A major limitation of the Boolean model is that it is not possible to obtain a ranking of the retrieval results. For example, if two documents satisfy the Boolean predicate of a query, the model does not specify which document to report first. This in turn does not conform to the user expectation of finding the document most likely to be relevant at the first rank, followed by those which are progressively less likely to be relevant. A second major disadvantage is that it may not be possible to reliably express the information need itself in a more complex than a simple Boolean predicate function. For example, a document containing the term ‘relativity’ may still be relevant to the query ‘relativity theory’ even if it does not contain the term ‘theory’.

## Vector Space Model (VSM)

The first generally adopted IR model was the vector space model (Salton et al., 1975). In VSM the each query  $q$  and document  $d$  is represented as a vector. If the vocabulary size is  $n$  then the dimension of both query and document vectors is also  $n$ . In the VSM, the similarity between a query and a document is computed as the cosine similarity between between the vector representations of the query and the document. The assumption in the VSM is that if a document is potentially relevant to a query, then the query and the document vectors will be *closer* in vector space (i.e. the angle between the query and document vector will be small). The similarity function for VSM is shown in Equation 2.1.

$$sim(d, q) = \sum_{i=1}^n d_i q_i \quad (2.1)$$

**Term Weighting.** The components of both query and document vectors are weighted in the VSM. The process of weighting components is known as *term weighting*. The term weighting mechanism in VSM depends on the following three factors.

1. **Term Frequency:** This is the number of times a particular term is present in a document or query. If a particular term is frequent in a document, it is reasonable to assume that the document is in fact one of those potentially relevant documents for the information need for the query containing this term. For example ‘proactive’ is a frequent term in this thesis. So for a query such as ‘proactive information retrieval’ this thesis could be a potentially relevant document candidate. If more weight is given to the word ‘proactive’, then for all queries comprising the term ‘proactive’ among others, this thesis will be ranked higher. However, the absolute count of term frequency is not found to be the most effective way of using the feature in retrieval (Singhal, 1997). For example, let there be two documents  $D_1$  and  $D_2$ . For the query be ‘web search paradigm’, let us assume that document  $D_1$  has only one word (i.e. ‘web’) that overlaps with the query. The term frequency of ‘web’ in  $D_1$  is 30. On the other hand, suppose  $D_2$  has two words (i.e. ‘web’ and ‘search’) overlapping with the query. To illustrate this point in our example, let the term frequencies of ‘web’ and ‘search’ in  $D_2$  are 5 and 10 respectively. In this scenario, it is more likely that  $D_2$  is more relevant to the query compared to  $D_1$ , because the word ‘web’ in  $D_1$  may also refer to ‘web design’ rather than ‘web search’. However using only the absolute term frequency count will rank  $D_1$  higher compared to  $D_2$ . So the term frequency function should give more importance to documents having a higher number of overlapping query terms. This is known as *coordination ranking* in the IR literature (Hiemstra, 2000).

The following are some commonly used term frequency functions from IR litera-

ture:

- the *augmented tf*:  $\frac{1}{2} + \frac{tf}{2_{\max}(tf)}$ , which normalizes the term frequency values within a range of  $[\frac{1}{2}, 1]$  (Salton and Buckley, 1988).
- the *logarithmic tf*:  $1 + \log(tf)$ , which is designed primarily to down-weight the contributions of terms with very high frequencies in a document.

It is easy to see that both the logarithmic and the augmented tf ensure that  $D_2$  is ranked higher than  $D_1$ . The score assigned to  $D_1$  by the logarithmic tf is  $1 + \log(20) = 1 + 2.99 = 3.99$ , whereas the score assigned to  $D_2$  is  $1 + \log(3) + 1 + \log(5) = 2 + 1.09 + 1.60 = 4.69$ , which is higher than that of  $D_1$ . The augmented tf in turn scores  $D_1$  as  $0.5 + 20/(2 \times 20) = 1$ , and  $D_2$  as  $0.5 + 3/(2 \times 5) + 0.5 + 5/(2 \times 5) = 1 + 0.3 + 0.5 = 1.8$ , which is higher than that of  $D_1$ .

2. **Inverse Document Frequency (IDF):** This estimates the importance of a term with respect to a collection (Salton and Buckley, 1988). If a term is one of the most frequent words in a document, that does not necessarily mean that the term is significant with respect to a collection of documents. Generally, common words (e.g. ‘the’, ‘and’) are removed from a collection since these words do not contain any topic-specific information in them. These are commonly called *stop-words* in the IR literature. However, a simple dictionary based filtering approach for removing stopwords may not be scalable and realized in practice, Firstly, because it depends on the availability of such a resource and how comprehensive it is, and secondly, for a purely collection frequency based hard filtering approach it may in fact be difficult to select an optimal threshold. To estimate the importance of those words in a document the concept of *idf* (inverse document frequency) was proposed. *idf* is inversely proportional to the number of documents in which a term is present. If a term is rare in a collection (i.e. it only appears in a proportionally low number of documents), then the presence of that term is important for a document. So, the *idf* value of a term depends on the

collection not on the particular document where it occurs. For example, suppose we are searching for a thesis on proactive information retrieval among all theses on information retrieval. In this example scenario, the term ‘proactive’ is likely to have a higher idf value compared to the other two terms, since each thesis in our collection is likely to contain the terms ‘information’ and ‘retrieval’ in its title. On the other hand, there would only be a small number of dissertations containing the word ‘proactive’, which would comprise the most likely candidate relevant documents pertaining to our query. The most commonly used formula for idf is  $idf(t) = \log(\frac{N}{df(t)})$ , where  $N$  is the total number of documents in the collection and  $df(t)$  is the number of documents in which  $t$  occurs (Sparck Jones, 1973).

3. **Document Length:** Generally speaking, the term frequency of a term is likely to be higher in longer documents compared to shorter ones. This does not necessarily mean that longer documents are more likely to be relevant for a query compared to a shorter one. Moreover, long documents are likely to have a greater number of query term overlaps compared to shorter documents. Hence length normalization was proposed to help mitigate the effect of document length on query-document matching scores.

There are different approaches for length normalization. One of them is cosine normalization where a document vector is normalized to unit vector by dividing each vector component with the total magnitude of the vector. This cosine normalization method applied on tf-idf vector produced by VSM approach did not perform well in early TREC collections (Singhal, 1997). This was later improved by applying pivot length normalization. In pivot length normalization, documents having length shorter than a threshold (i.e.  $l_t$ ) is boosted and similarly documents having length greater than a threshold (i.e.  $l_t$ ) are down weighted. The threshold length (i.e.  $l_t$ ) is obtained by training on a set of queries and relevance judgements (Singhal, 1997).

Both the term frequency ( $tf$ ) and the inverse document frequency ( $idf$ ) are important in creating effective term weights for retrieval, it is a standard practice in the VSM to combine these components simply by multiplying them together to capture both their effects. This combination is commonly known as  $tf-idf$  weighting.

The major criticism against the VSM is that the model itself is not derived from a theoretically sound basis for determining the term weighting components, such as which  $tf$  function to use, whether to use cosine normalization or the pivoted length normalization etc. Neither does the VSM model theoretically justify the multiplicative combination of  $tf-idf$  weighting. The VSM has subsequently been followed by the development of more theoretically motivated IR models which we review next.

## Probabilistic Model

The first major alternative is the probabilistic model (Robertson, 1977). This seek to estimate the probability of a document being relevant to given a query ( i.e.  $P(d = R|q)$ ). The documents in a ranked retrieval list are ordered based on decreasing  $P(d = R|q)$ .

One version of the probabilistic model uses binary independence model (BIM). In BIM, the assumption is that terms are pairwise independent. The limitation of BIM approach is that it depends only on the presence of a term in a document and does not take into account factors like document length and term frequency.

The limitations of the basic BIM were addressed in the BM25 weighting model was proposed (Robertson et al., 1994; Sparck-Jones et al., 2000). Equation 2.2 shows the scoring formula for BM25 for a document  $d$  and a query  $q$ .

$$sim(d, q) = \sum_{t \in q} \log \frac{N}{df(t)} \times \frac{(k_1 + 1)tf(t, d)}{k_1(1 - b + b\frac{L_d}{L_{avg}}) + tf(t, d)} \quad (2.2)$$

In Equation 2.2,  $tf(t, d)$  denotes the term frequency of a term  $t$  in a document  $d$ ,  $L_{avg}$  is the average length of a document in the collection,  $L_d$  is the length of a

document  $d$  and  $df(t)$  is the document frequency for a term  $t$ . The effect of the tuning parameters  $k_1$  and  $b$  are described below.

- $k_1$  controls the contribution of term frequency in similarity computation. At  $k_1 = 0$ , similarity depends only on the idf value. The higher the value of  $k_1$ , the higher is the impact of term frequency on the overall BM25 weight.
- $b$  ( $0 \leq b \leq 1$ ) controls the effect of length normalization.  $b = 1$  indicates full length normalization and  $b = 0$  indicates no length normalization.

The values of  $k$  and  $b$  are set empirically in a training phase.

### Language Modeling (LM)

A later retrieval model is the language modelling (LM) (Ponte and Croft, 1998; Hiemstra, 2000). The LM approach computes the probability of generating a query given a document. The assumption in LM is that each query term is sampled from a document or from the collection. Sampling a query term from a document is analogous to the use of term frequency of a query term as the estimation of importance of the document with respect to the query term. Similarly sampling a query term from a collection is analogous to the use of idf as the estimation of importance of the document with respect to the query term.

In the LM approach, documents are sorted based on the decreasing values of posterior probabilities of generating a document  $d$  given a query  $q$ , i.e.  $P(d|q)$ . The derivation for  $P(d|q)$  is given in equation 2.3. Since the denominator of Equation 2.3 is constant,  $P(d|q)$  can be treated as simply proportional to  $P(q|d) * P(d)$ .

$$P(d|q) = \frac{P(q|d)P(d)}{\sum_{d' \in C} P(q|d')P(d')} \quad (2.3)$$

$P(q|D)$  can be computed using Equation 2.4.

$$P(q|d) = \prod_{t \in q} \lambda P_{MLE}(t|d) + (1 - \lambda) P_{coll}(t) \quad (2.4)$$

In Equation 2.4,  $P_{MLE}$  is the maximum likelihood estimate of generating a query term  $t$  from a document  $d$ . Calculation of  $P_{MLE}$  is shown in Equation 2.5.

$$P_{MLE}(t|d) = \frac{tf(t, d)}{L_d} \quad (2.5)$$

In Equation 2.4,  $P_{coll}(t)$  is the probability of generating the term  $t$  from the collection. Calculation of  $P_{coll}(t)$  is shown in Equation 2.6. In Equation 2.6,  $df(t)$  is the number of documents in which the term  $t$  occurs.

$$P_{coll}(t) = \frac{df(t)}{\sum_{t'=1}^n df(t')} \quad (2.6)$$

In some cases, collection frequency (i.e.  $cf(t)$ ) is used to compute  $P_{coll}(t)$ . collection frequency is the number of times a term  $t$  occurs in the collection, which is actually similar to document frequency. The variable  $cf(t)$  is normalized by the total number of terms in the collection.

As described earlier, the working principle of the LM approach depends on two events: sampling a term from a document and sampling a term from a collection. In Equation 2.4,  $\lambda$  and  $1 - \lambda$  are the probabilities of the two events.  $\lambda$  is similar to  $k_1$  in BM25, since it balances between the two events. If a query term  $t$  is not present in a document  $d$ , then  $P(t|d)$  is not zero because  $\lambda$  adds the contribution of  $P(t|C)$  in  $P(t|d)$ . For this reason,  $\lambda$  is commonly known as a smoothing parameter.

In Equation 2.3,  $P(d)$  can take account of the document length factor, since  $P(d)$  can be a function of the document length of  $d$ , denoted by  $L_d$  (Hiemstra, 2000).

The work in (Hiemstra and Kraaij, 2005) showed that an LM approach with non-uniform document length priors can outperform the BM25 retrieval model by 8.19% on the TREC-7 dataset.

The issue in the language model proposed in (Hiemstra and Kraaij, 2005) is that there may be vocabulary mismatch between queries and documents. In this scenario exploitation of the semantic relations between words can be an effective means to



improve the language model. In this regard we can broadly categorize two different threads of work. One is related to Latent Dirichlet Allocation (LDA) based document language models proposed in (Wei and Croft, 2006). In this work the assumption is that a document collection contains multiple topics. Each document is represented as a distribution of different topics. Similarly each word is also represented by a distribution of topics. The presence of topics in the representation of words and documents helped to capture semantic relation between words to improve retrieval effectiveness.

A different thread of work for capturing word semantics used translation language models (Karimzadehgan and Zhai, 2012, 2010). In translation based language models, for each word the translation probabilities into the query words are estimated. The work in (Karimzadehgan and Zhai, 2012, 2010) used document level word-cooccurrences to estimate the translation probabilities. Retrievals based on translation language models (Karimzadehgan and Zhai, 2012) outperformed basic language model (Hiemstra and Kraaij, 2005) across several TREC collections.

We now turn our attention to relevance feedback models.

### 2.1.2 Relevance Feedback Models

Relevance feedback models (RF) seek to improve the effectiveness of retrieval models. In RF, initial search queries are modified or query terms can be re-weighted. An IR system uses the feedback documents to refine the search with adjusted parameters aiming to eventually return a modified ranked list containing more relevant documents at better ranks. Relevance feedback can be applied in two ways:

- *Query term reweighting*: This involves reweighting the terms of the query. The objective of term reweighting is to increase the weight of terms that are likely to be relevant and to down-weight those which are likely to be non-relevant to the query.
- *Query expansion (QE)*: In this case additional terms, i.e. terms which do not

already appear in the current query, are added from the feedback documents. Typically, QE is accompanied by the reweighting of the query terms.

RF can be based on true relevant documents or pseudo relevant documents. Generally speaking, users are not keen to provide feedback to a search system. However, obtaining RF information from real users is also not possible in completely automatic IR since only the searcher can determine true relevance. Consequently, methods of implicitly obtaining feedback are common in practice. Implicit feedback can either be obtained through user interaction events such as clicks and subsequent document visiting times, or by simply assuming that a certain number of top ranked documents from an initial retrieval step are relevant. This latter technique is known as pseudo relevance feedback (PRF), which is a simple and often effective technique to improve on the initial retrieval output in the absence of explicit user feedback.

### Classical Relevance Feedback Methods

**Rocchio Relevance Feedback.** A classic example of term reweighting for PRF is provided by the Rocchio RF method (Rocchio, 1971) developed in relation to the VSM. Recall that in the VSM, both the query and the documents in the collection are represented as vectors. The relevant or pseudo-relevant document set available for RF is thus a set of vectors. The objective of the Rocchio method is to shift the query vector towards the centroid of these relevant vectors and away from the centroid of the non-relevant ones. This shifting of the query vector is realized by reweighting its components. The query modification algorithm as proposed by Rocchio is shown in Equation 2.7. The parameters  $\alpha$ ,  $\beta$  and  $\gamma$  are the scalar constants attached to the original query vector  $q$ , the set of judged relevant documents in the feedback step ( $R$ ), and the complementary set of non-relevant documents ( $NR$ ) respectively. The values of  $\alpha$ ,  $\beta$ , and  $\gamma$  are set empirically for the current retrieval task.

$$q' = \alpha q + \frac{\beta}{|R|} \sum_{d \in R} d - \frac{\gamma}{|NR|} \sum_{d \in NR} d \quad (2.7)$$

Apart from re-weighting query terms using Equation 2.7, Rocchio’s method gives us a mechanism for query expansion. Any word appearing in the set of relevance feedback documents (i.e.  $R$ ) are potential candidates for expansion terms. The weight of the expansion term can be estimated using Equation 2.7. Once we have weights for each potential expansion term then we choose top  $k$  terms to expand the original query.

### **Relevance Feedback with the Robertson/Sparck Jones Relevance Weight.**

For the probabilistic model, the most commonly used method to reweight query terms is based on the Robertson/Sparck Jones relevance weight (RW) (Robertson et al., 1994), shown in Equation 2.8.

$$RW(t) = \log \frac{(r + 0.5)/(R - r + 0.5)}{(n - r + 0.5)/(N - n - R + r + 0.5)} \quad (2.8)$$

In Equation 2.8,  $r$  is the number of known relevant documents in which the term  $t$  occurs,  $N$  is the total number of documents in the collection,  $n$  is the total number of documents in which term  $t$  occurs, and  $R$  is the number of known relevant documents. The objective of the RW score is to put more emphasis on terms with high idf values which occur frequently in the (pseudo-)relevant documents.

Similar to (Rocchio, 1971), probabilistic model also provides means for query expansion. Potential expansion terms again come from the relevance feedback documents. Potential expansion terms are ranked using the Robertson Offer Weight (Robertson, 1990) shown in Equation 2.9. In Equation 2.9,  $r$  is the number of relevant documents where a term  $t$  occurs and  $RW(t)$  is the Robertson weight obtained using Equation 2.8.  $OW(t)$  in Equation 2.9 is termed the *offer weight*. The top  $k$  terms computed using  $OW(t)$  are added to the original query.

$$OW(t) = r * RW(t) \quad (2.9)$$

## Relevance Language Model

The relevance model (RLM) is based on the assumption that the relevant documents corresponding to a query are sampled from a latent relevance model  $R$ . The RLM seeks to estimate  $R$ . Since it is not possible to know the relevant documents for a query beforehand, the top retrieved documents for a query  $Q$  are assumed to be relevant for that query. The probability of sampling a term  $t$  from  $R$  is denoted by  $P(t|R)$ . Since  $Q$  is the only evidence about the relevance model,  $P(t|R)$  is approximated using  $P(t|Q)$ . Let  $Q$  be comprised of words  $q_1, q_2, \dots, q_k$ . We show the definition of  $P(t|Q)$  in Equation 2.10.

$$P(t|Q) = \frac{P(t, q_1, q_2, \dots, q_k)}{P(q_1, q_2, \dots, q_k)} \quad (2.10)$$

$P(t, q_1, q_2, \dots, q_k)$  can be estimated using two different approaches: IID sampling and conditional sampling. Each of these is described below.

**IID Sampling.** Here the assumption is that  $q_1, q_2, \dots, q_k$  and  $w$  are sampled identically and independently from a unigram distribution  $M_R$ . Let us assume that  $M$  represents the finite set of unigrams in the universe. The sampling process chooses a  $M_R$  from  $M$  and then tries to estimate the probability  $P(t, q_1, q_2, \dots, q_k|M_R)$ . This continues for  $K + 1$  terms. This process is shown in Equation 2.11.

$$P(t, q_1, q_2, \dots, q_k|M_R) = P(t|M_R) \prod_{i=1}^k P(q_i|M_R) \quad (2.11)$$

Eventually  $P(w, q_1, q_2, \dots, q_k)$  is computed using Equation 2.12.

$$P(t, q_1, q_2, \dots, q_k|M) = \sum_{M_R \in M} P(t, q_1, q_2, \dots, q_k|M_R) \quad (2.12)$$

**Conditional Sampling.** In this sampling strategy, each query word is sampled independently, but the term  $w$  is dependant on each query term. More formally, we

estimate  $P(w, q_1, q_2, \dots, q_k)$  as shown in Equation 2.13.

$$P(t, q_1, q_2, \dots, q_k) = P(t) \sum_{i=1}^k P(q_i|t) \quad (2.13)$$

To compute  $P(q_i|t)$ , we estimate both the probability of observing a unigram distribution  $M_R$  given the word  $t$ , and the probability of observing a query word  $q_i$  given  $M_R$ . Equation 2.14 describes the mathematical formulation of  $P(q_i|w)$ .

$$P(q_i|t) = \sum_{i=1}^k P(M_R|t)P(q_i|M_R) \quad (2.14)$$

The two variants of RLM in Equations 2.11 and 2.13 are referred to as ‘RM1’ and ‘RM2’, respectively. The RLM, as proposed in, does not consider the original query terms for estimating the density function. The work in proposed two new variants ‘RM3’ and ‘RM4’ for existing relevance language model. The process of computing  $P(t|Q)$ , for RM3 and RM4 is shown in Equations 2.15 and 2.16 respectively.

$$P(t|Q) = (1 - \lambda)P_{MLE}(t|Q) + P_{RM1}P(t|Q) \quad (2.15)$$

$$P(t|Q) = (1 - \lambda)P_{MLE}(t|Q) + P_{RM2}P(t|Q) \quad (2.16)$$

As shown in Equations 2.15 and 2.16, the query model  $P(w|Q)$  is computed using both maximum likelihood  $P_{MLE}(t|Q)$  estimation (MLE) and one variant of RLM. Since RM3 has been shown to empirically outperform RM4 (and also RM1 and RM2, i.e., the respective models without the query mixture components), we use the RM3 version of RLM as the baseline for our experiments, and simply refer to it as RLM throughout the rest of the thesis.

## **Kernel Density Based Relevance Language Model (KDERLM)**

The RLM focuses only on word-cooccurrence to estimate the relevance of a word. The KDERLM was proposed to incorporate word embedding based semantic sim-

ilarity into relevance language model. Kernel Density Estimation (KDE) is a non-parametric method to estimate the probability density function of a random variable. Formally, let  $\{x_1, x_2, \dots, x_n\}$  be independent and identically distributed (i.i.d.) samples drawn from a distribution. The shape of the density function,  $f$ , from which these points are sampled can be estimated is described in Equation 2.17. In Equation 2.17,  $x_i$  is a given data point (commonly known as the pivot point),  $\hat{f}_\alpha(x)$  is the estimated value of the true density function  $f(x)$ ,  $\alpha_i$  is the relative importance of the  $i^{th}$  data point with the constraint that  $\sum_{i=1} \alpha_i = 1$ , and  $K(\cdot)$  is a kernel function scaled by a bandwidth parameter  $h$ .

$$\hat{f}_\alpha(x) = \frac{1}{nh} \sum_{i=1}^n \alpha_i K\left(\frac{x - x_i}{h}\right) \quad (2.17)$$

By definition, a kernel function is a monotonically increasing function of the distance between two points (vectors). A common choice of kernel function is a Gaussian function.

(Roy et al., 2016) observed that since the relevance model (Lavrenko and Croft, 2001) estimates a distribution of (real-valued) weights over terms, the concept of KDE can be applied to define this distribution in a generalized way (the model being called Kernel density based RLM or KDERLM for short). The basic idea to define the relevance model distribution this way is to treat the query terms as a set of pivot terms (analogous to the  $x_i$ 's of Equation 2.17). Rather than treating terms as independent, the distance between the vector representations (obtained by applying a word embedding method such as word2vec (Mikolov et al., 2013a)) of a pivot (query) term with that of a term occurring in the top-ranked documents is then used to define the kernel function. This results in the influence of a query term to propagate to other terms that have similar (close) vector representations in the embedded space. Formally, assuming that the query terms  $Q = q_1, \dots, q_n$  are embedded as vectors, the probability density function estimated with KDE is

described in Equation 2.18.

$$f(t) = \frac{1}{nh} \sum_{i=1}^n P(t|M)P(q_i|M)K\left(\frac{t-q_i}{h}\right) \quad (2.18)$$

In Equation 2.18, the the kernel function  $K$  is a function of the distance between the word vectors of a term  $w$  (within a top-ranked document) and a query term  $q_i$ . Moreover,  $P(t|M)P(q_i|M)$  acts as the weight associated with this kernel function (thus incorporating the local RLM effect in addition to the global term semantics one from the embedded space). In other words, the closer the word  $w$  is to a query term  $q_i$  in conjunction with a high RLM term weight, the higher becomes the value of the KDERLM weight  $f(t)$ .

## 2.2 Proactive Information Retrieval

In existing state-of-the-art IR models, a user has to enter a query to locate potentially relevant information sources. One of the major factors for an IR model to perform well is the expertise of the user in query formulation. In general a query which is too long, or a query with less important words in it, does not perform well in an IR model. To reduce a user’s effort of typing queries, and to minimize the effect of user’s expertise in query formulation, existing literature has proposed different ways to make an IR model proactive. Existing work can be categorized into two different threads: methods to find semantic associations among different desktop items to enhance desktop search; alternative proactive suggestion algorithms. Each of these is described below.

### 2.2.1 Personal Information Management (PIM)

In PIM the focus is on organising the diverse information sources of an individual user (e.g. files, emails, bookmarks) to support the user in their work. In this thread of work methods for creating semantic representations of desktop items have been

examined in a number of studies including (Chen et al., 2009; Kim et al., 2010; Bernstein et al., 2007; Cai et al., 2005).

The study in (Cai et al., 2005) proposed an interface architecture named ‘SEMEX’ which represented each of the items accessed by the user in desktop as objects and captured the relation between different objects based on their domain and the user’s interaction pattern. They showed that the ‘SEMEX’ architecture could help a user in browsing his desktop data in a more effective way. The work in (Bernstein et al., 2007) conducted a user study to investigate how short and self-contained notes known as *information scraps* could help to capture a user’s thought process at different stages of his interaction with a computational device. The work in (Chen et al., 2009) proposed an associative memory based system named ‘iMecho’ which sought to enhance a traditional desktop search system. iMecho finds semantic connections among desktop resources exploiting a user’s activity pattern. The semantic links help to anticipate the context of a user’s mind while using a desktop search system. Moreover, it also improved the ranking of retrieved items for personalized desktop search by taking into account both relevance and importance of a desktop item to the user. In addition, iMecho provides a faceted search feature and association graph navigation to help users refine and associate search results generated by full-text keyword search. The iMecho prototype showed that an association-based search system is better than traditional keyword search for desktops, since it is closer to the way that human associative memory works.

The work in (Kim et al., 2010) proposed a methodology to create a semantic representation of personal information for a user. A typical collection of personal information contains many documents and mentions many concepts (e.g., person names, events, etc.). Kim et al. (2010) used click feedback to create semantic associations between items and concepts and showed that semantic representation is useful for a known item search task in a desktop setting. They also showed that the system can learn to predict such associations with a small amount of click data.

The study in (Hu and Janowicz, 2012) first attempted to integrate information



relating to a user’s physical activities (e.g. paper writing, conference arrangements) with information associated with their digital footprint (e.g. accessing adobe reader, conference related papers, latex) to improve their search experience when using their desktop environment. In this work an activity ontology for a user’s conference planning task was developed. The activity categories related to the conference planning task were ‘paper writing’, ‘conference arrangements’ and ‘conference participation’. This work showed that combining physical activity information with digital footprints improved the desktop search system. The enhanced desktop search system provided results for complex search queries such as ‘which papers from the ACM GIS proceedings have I opened during the conference?’

A major difference between our work and existing PIM related work is that whereas in this previous work researchers focused on improving the search results for a user query, in our PIR work the focus is on predicting what a user may need in the future. This is related to the scenario where a user has not actively entered a query and the objective is to find what information sources may be useful to the user at that point, or to the situation where a user has entered a query and the objective is to predict what further information needs might arise in his mind.

### 2.2.2 Proactive Suggestion Methods

The first and the oldest work on proactive suggestions was proposed in (Rhodes, 2000). This work introduced the term ‘remembrance agent’ (RA) to describe their prototype proactive suggestion system which was deployed in a UNIX text editor named EMACS. The RA was used to provide suggestions to the user based on what they were reading or writing. Similarly, the study in (Rhodes and Maes, 2000) proposed *just-in-time* proactive retrieval agents which also provided proactive suggestions based on context information of the user’s current desktop activities on the web or using a text editor. Over the years the context used to provide proactive suggestions has become more extensive (Vuong et al., 2017).

The work in Dumais et al. (2004) proposed a prototype system capable of gen-

erating document suggestions for a user based on their recent desktop activities. A query is formulated using the last words typed by a user and the query is then used to identify potentially interesting documents for the user. They conducted a user-centric evaluation based on the number of clicks by the user on document notifications within a controlled setting of email composition to measure the effectiveness of the proactive suggestion system.

Another system ‘Stuff I’ve Seen’ (SIS (Dumais et al., 2016) seeks to improve the desktop search experience of a user. Stuff I’ve Seen facilitates information reuse in desktop systems. It constructs a unified index of information that a person has accessed previously, e.g. emails, web pages, documents, appointments, etc., and then carries out desktop search by leveraging the user’s current activity context to formulate and expand queries.

A similar thread of work identifies documents related to a task from a user’s past activities by taking into account different factors, such as access frequency of documents, difference between the most recent access time of a document and the present timestamp (Tran et al., 2016). A drawback of the study reported in (Tran et al., 2016) is that it is limited by the need for the presence of user-assigned tags of desktop items to estimate semantic relations between them.

The ‘accessrank’ algorithm proposed in (Fitchett and Cockburn, 2012) predicts which item a user is likely to access next in a desktop environment by making use of the access frequency of the items. A drawback of the proposed algorithm is that it mostly works well in scenarios where a user frequently revisits a previously accessed item.

Recently the work in (Shokouhi and Guo, 2015) proposed a method for re-ranking proactive cards relating to a user based on different factors, including queries typed by the user in past, the user’s location, and the time of the day. Information cards in this scenario are information snippets related to topics in the user’s current interaction context on a mobile device. In their study the authors automatically annotated each document accessed by the user with a category described in the

Open Directory Project <sup>1</sup>. Given the annotations, the proposed model suggested a list of information cards to the user based on the topic in which the user was interested at that moment. In a similar thread of work, (Yang et al., 2016) used features such as entity tagging and user demographic information to re-rank the information cards. The work in (Song and Guo, 2016) investigated different types of day-to-day task (e.g. weather check, stock market prediction check) that a user generally repeats, to improve proactive recommendation of information cards. The difference between our work and (Shokouhi and Guo, 2015; Yang et al., 2009; Song and Guo, 2016) is that rather than finding a user’s general topic of interest at a point in time from their interaction log, we focus on identifying the broader level task which the user seeks to accomplish at a particular point in time.

Surveillance of digital activities (e.g. emails, messaging, searching in web etc.) of a user was shown to be useful for proactive suggestions (Vuong et al., 2017). A relatively simple simulated user read activity involving selection of top tf-idf terms from a document was proposed in (Koskela et al., 2018). An apparent weakness of this simple approach is that, a set of top tf-idf terms are most likely to be non-contiguous and hence are likely not to reflect a realistic user behaviour of reading a document. The study reported in (Koskela et al., 2018) also described a controlled user study of essay writing on a specific set of topics with real users, and showed that a proactive retrieval system, which allows a user to read suggested documents or save them for later revisits, can help a user to accomplish his task more efficiently.

Application of topic models on user accessed documents has been demonstrated to enable the development of more proactive search interfaces to minimize user query formulation effort (Vuong et al., 2017; Ganguly et al., 2013). The work in (Van Gysel et al., 2017) proposed a method for proactively suggesting email attachments based on email content. The study in (Hinbarji et al., 2016) showed that continuously storing the digital content presented on a screen can act as a memory storage of a user’s activity patterns. This store can then potentially be useful if a user needs to

---

<sup>1</sup><http://www.dmoz.org/>

look back or recall any particular activity to assist them in completing a task in the future.

Most of the work on proactive IR introduced above either conducted user studies (Dumais et al., 2016; Koskela et al., 2018) to explore proactive suggestion algorithms or used annotated a dataset (Tran et al., 2016) for proactive suggestion. In contrast, our objective uses a reproducible framework where we can compare different approaches, which do not depend on having manual annotations carried out by the user himself.

### 2.2.3 Query Suggestion Methods

This work is generally intended to support a user in formulating or reformulating his search queries by enabling him to select their next query from a list of potential candidates. In a web search setup, the search engine generally uses contextual information from the user’s current or previous search sessions to improve the search effectiveness. The work in (Smyth et al., 2003) used similar queries of other users to re-rank search results for the current query of a user. In similar work in (Smyth et al., 2009) they proposed a novel architecture for a search interface to support a user in his current search session by leveraging similar task information from other users. For each user they provided an option to manually annotate the task in which the user was currently engaged. The task information was stored in *search staks*. The system included an option to share the *search staks* across different users so that collaborative information could be used to support a current user in his task. The difference between our proactive suggestion approach and this existing work in (Smyth et al., 2009) is that rather than using manual task annotation, our object was to automatically mine user tasks from a search query log. The study in (Feild and Allan, 2013) analyzed the importance of a user’s previous on-task queries (related to the current task) compared to off-task (not related to the current task) for query recommendation. An extended relevance model that can leverage a user’s previous queries within a session feedback signals for re-ranking the top retrieved

documents was proposed in (Levine et al., 2017). Similarly, Zhang et al. (2013) used edit distances between pairs of adjacent queries during search sessions as a pseudo feedback signals to improve retrieval effectiveness of the most recent query.

In addition to using the context information from recent queries, other approaches investigated improving retrieval effectiveness by leveraging information from a user’s long-term search history (query logs), e.g., (Tan et al., 2006) estimates a query-focused user profile from a user’s past interactions (queries, clicked documents) that are similar to his current query, which is then used to re-rank the documents for the current query.

On a similar note, the studies in (Kong et al., 2015; Sordoni et al., 2015) use a combination of information from current session and historical interactions to recommend a list of suggested queries, whereas (Boldi et al., 2008) constructs a ‘query-flow’ graph from a query log to predict a candidate list of next queries.

A task graph was employed in (Hassan Awadallah et al., 2014) to suggest a a query on a sub-task different from the current one he is focusing on. (Muntean et al., 2013) applied ‘learning to rank’ to predict the final query in a search session aiming to automatically stop query suggestions to increase user satisfaction. The study in (Li et al., 2012) seeks to introduce diversity into query suggestions by making use of the historical information of the user interactions (queries and clicked documents).

A visualization tool that shows connections between the documents retrieved for a user’s current query and the ones that were retrieved for his previously submitted queries during the different stages of a task was shown to improve exploratory search experience in (Qvarfordt et al., 2013).

The difference between the query suggestion or query prediction methods described above and our PIR approach is that our objective is to support a user in his overall task (which may have one or more sub-tasks in it) rather than focusing only on the next information need of the user (Zhang et al., 2013; Levine et al., 2017; Sordoni et al., 2015).

## 2.3 Representation Learning for Text

In Chapter 1 we introduced research question RQ2 which relates to anticipating a user's information need by first identifying activities related to their information goal. Generally speaking, we need to identify the semantic relations between a pairs of activities. In order to find related activities, in this study we focus on activities expressed only in text. In this section we first discuss language modeling techniques which attempt to capture the word semantics at the surface level of word character units, and after this we introduce some recent work which seeks to model semantic relations between text units via word embedding methods.

### 2.3.1 Language Modeling Techniques for Word Semantics

One of the earliest proposed ways of capturing word semantics was the use of statistical language models. In this models, probabilistic methods are used to estimate the likelihood of a word given a context of other words. One of the earliest uses of statistical language models was in the field of automatic speech recognition (Bahl et al., 1983), to assist in correctly recognizing words and phrases in sound signals that have been subjected to noise and/or faulty channels. While a full probabilistic model containing the likelihood of every word, given all possible word contexts that may arise in a language is clearly intractable, it has been empirically observed that satisfactory results are obtained using a context size as small as 3 words (Goodman, 2001). A simple mathematical formulation of such an  $n$ -gram model with context window size equal to  $W$  follows:

$$P(w_1^W) = \prod_{t=1}^W p(w_t | w_1^{t-1}) \quad (2.19)$$

In Equation 2.19,  $w_t$  is the  $t^{th}$  word and  $w_i^W$  refers to the sequence of words from  $w_i$  to  $w^T$ , (i.e.  $(w_i, w_{i+1}, w_{i+2} \dots w_W)$ ).  $P(w_t | w_1^{t-1})$  refers to the fraction of times  $w_t$  appears after the sequence  $w_1^{t-1}$ . Actual prediction of the next word given a context

is done via maximum likelihood estimation (MLE), over all words in the vocabulary.

The study in (Yoshua Bengio and Jauvin, 2003) reported some issues with the approach proposed in (Bahl et al., 1983). High dimensionality is involved in calculating discrete joint distributions of words with vocabulary sizes of the order of 100,000 words for the approaches proposed in (Bahl et al., 1983). Early attempts at mitigating these issues, particularly those related to generalization to unseen phrases, include the use of smoothing, e.g. pretending every new sequence has count one, rather than zero in the training set (this is referred to as add-one or Laplace smoothing). There is still a problem to apply statistical language modeling on dataset having large vocabulary.

To alleviate this issue, neural networks (Yoshua Bengio and Jauvin, 2003; Bengio and Senécal, 2003) and log-linear models (Mnih and Hinton, 2007; Mikolov et al., 2013a) were introduced to train language models (giving rise to so called neural language models). Neural language models (NLM) have fixed dimensional dense representation for words and thus the dimension of embedding does not depend on the size of vocabulary. As a result of this, NLMs address the problem of modelling large vocabularies. The method of obtaining word representations in terms of fixed length vectors is known as *word embedding*. Since in our research scope, we use word embedding to capture the semantics of user activities, in the following section we review key existing literature on NLMs which are used for word embedding.

### **2.3.2 Word Embedding: A Generic Survey**

Word embedding is the process of obtaining a vector representation of a word. The word vector is supposed to capture the semantic meaning of the word. For example if we have the word vectors for three words namely ‘dog’, ‘cat’ and ‘tree’, then the similarity between the word vectors for ‘dog’ and ‘cat’ should be more compared to that of ‘dog’ and ‘tree’ or ‘cat’ and ‘tree’. This section introduces various approaches to training word embeddings, detailing how they work and where they differ from each other. Word embeddings are commonly ((Baroni et al., 2014; Socher et al.,

2011; Li et al., 2015)) categorized into two types, depending upon the strategies used to induce them. Methods which leverage local data (e.g. a word’s context) are called *prediction based models*, and are generally reminiscent of neural language models. On the other hand, methods that use global information, generally corpus-wide statistics such as word counts and frequencies are called count based models. Since our work involves constructing word vectors based on its context, we describe the prediction based models in detail in the following subsection.

### Prediction Based Models

The history of the development of prediction based models for embeddings is deeply linked with that of neural language models (NNLMs), because that is how they were initially produced (Almeida and Xexéo, 2019) (Yoshua Bengio and Jauvin, 2003). The first large neural language model (Yoshua Bengio and Jauvin, 2003) is mostly one of gradual efficiency gains, occasional insights and trade-offs between complex models and simpler models, which can be trained on more data. Early results clearly indicated that NNLM are better at modelling language than their previous n-gram-based counterparts. Moreover, long training times (e.g. days and weeks) are frequently cited among the major factors have that affected the development of such models.

The study in Bengio and Sen  cal (2003) identified that one of the potential sources of computational cost was the partition function or normalization factor required by softmax output layers, such as those in NNLMs. Using a concept called importance sampling (Doucet (2001)) reported gains of a factor of 19 in training time, with respect to the previous model, with similar scores (as measured by perplexity). (Morin and Bengio, 2005) proposed yet another approach for speeding up training and testing times, using a Hierarchical Softmax layer. They realized that, if the output words are arranged in a hierarchical binary tree structure, then the probability distribution of each word can be calculated using the probability that each node leads to the word using the correct path from the binary tree. Since the



height of a binary tree over a set  $V$  of words is  $|V|/\log(|V|)$ , this could yield exponential speedup. In practice, gains were less pronounced, but they still managed gains of a factor of 3 for training times and 100 for testing times, with respect to the model using importance sampling.

Mnih and Hinton (2007) suggested the Log-bilinear Model (LBL) which has been very popular in later works. An extension of the LBL (Mnih and Hinton, 2007) is (Mnih and Hinton, 2009). The work in (Mnih and Hinton, 2009) used a slightly modified version of the hierarchical softmax scheme proposed by (Morin and Bengio, 2005)), yielding a Hierarchical Log-bilinear Model (HLBL).

Somewhat parallel to the works just mentioned, (Collobert and Weston, 2008) approached the problem from a slightly different perspective. They first designed a model with the specific intent of learning embeddings only. In previous models, embeddings were just treated as an interesting by product of the main task (usually language models). In addition to this, they also introduced two improvements: they used a word’s full context (before and after) to predict the centre word. They also introduced a more sophisticated way of leveraging unlabelled data for producing good embeddings instead of training a language model (which is not the eventual task for term semantics), they expanded the dataset with false or negative examples and simply trained a model that could tell positive (actually occurring) from false examples. Here we should mention two specific contributions by (Mikolov et al., 2009), which have been used in later models. In the first work, (Mikolov et al., 2009) a two-step method for bootstrapping a NNLM was suggested, whereby a first model was trained using a single word as context, and then a second model (with a larger context) was trained, using as initial embeddings those found by the first step.

It could be said that, in 2013, with (Mikolov et al., 2013a) the NLP community was interested in word embeddings as a topic worthy of research in and of itself. Since we use Word2vec (Mikolov et al., 2013a) extensively in our work for proactive suggestions, in the following section we will describe in detail the word2vec model

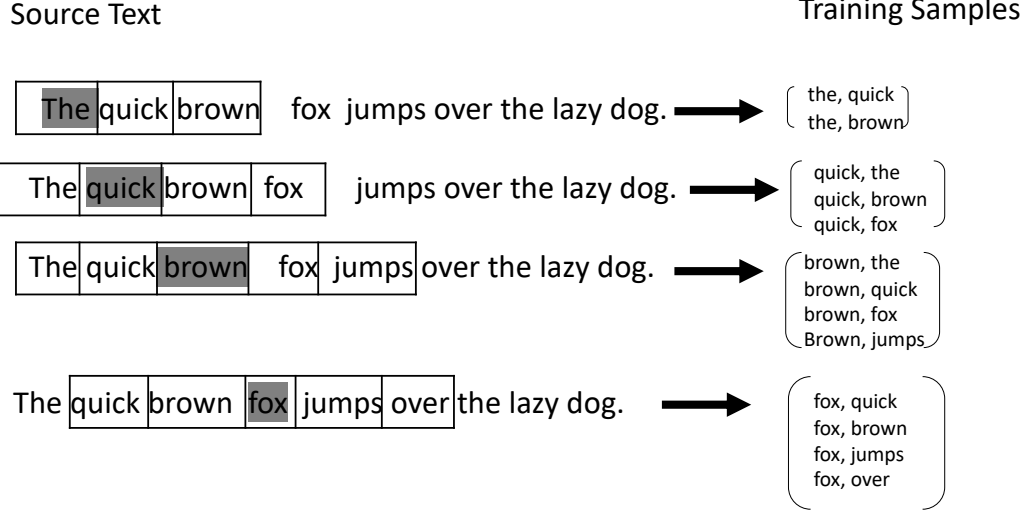


Figure 2.2: Example of the use of sliding window in computation of word embedding using word2vec

proposed in (Mikolov et al., 2013a).

### 2.3.3 Word2vec Model Overview

Word2vec (Mikolov et al., 2013a) is a weakly supervised technique to obtain word vectors. Here we slide a window through unlabeled text. The objective of Word2vec is to make the current word similar to the all words appearing within its context window, and to make the current word dissimilar to randomly sampled words outside its context. Figure 2.2 shows an overview of principles of the word2vec model. For each word, it considers a context window of size four, two on each side. For example, for the word ‘fox’ it tries to make it similar to jump, over, quick and brown. The context window for word2vec does not cross the document boundaries. The objective function for word2vec is described in Equation 2.20. The objective function for Word2Vec is maximized using stochastic gradient descent.

$$J(W) = \sum_{w_t, c_t \in D^+} \sum_{c \in c_t} P(D = 1 | w_t, c) - \sum_{w_t, c'_t \in D^-} \sum_{c \in c'_t} P(D = 1 | w_t, c) \quad (2.20)$$

## Neural Architecture

In Word2vec, the input to the model is either a word or a word context window of a fixed size. Each input word is represented as a one-hot vector. The dimension of the vector is equal to the number of unique words present in the vocabulary. For example, for the word ‘brown’, there is “1” in the position corresponding to the word ‘brown’, and ‘0’s in all of the other positions. Similarly, the context vector is also represented as the concatenation of the one hot vectors of the corresponding word vectors. The output of the network can be either a context or a single word.

The word2vec model which takes as input a context vector and predicts the word is known as CBOW model. Similarly the word2vec model which takes as input a word and predicts its context is known as Skipgram model. Both model is trained on a simple neural network with a single hidden layer to perform a certain task. Ultimately the neural network for the task on which it was trained is not used. During training of the networks, it actually learns the weights of the hidden layer. The weights are the word vectors. There is no activation function on the hidden layer neurons, but the output neurons use sigmoid.

Figure 2.3 shows the neural network architecture for skipgram in Word2vec. It is shown in Figure 2.3 that given the word  $w_t$ , it predicts a context window of size 4 (i.e.  $W(t-1), w(t-1), w(t+1), w(t+2)$ ). The output vector is actually a probability distribution through a sigmoid function.

### 2.3.4 Contextual Word Embedding

As an alternative to the singleton word embedding methods discussed in Section 2.3.2, more recently researchers have proposed contextual word embedding methods. The main difference between this and the singleton methods is that these involve estimating vectors for each word within a given context (e.g. a sentence), which also implies that the vector for a word mostly varies in different contexts, e.g. the vector for the word ‘school’ in the sentences ‘John’s dad takes him to school’ and

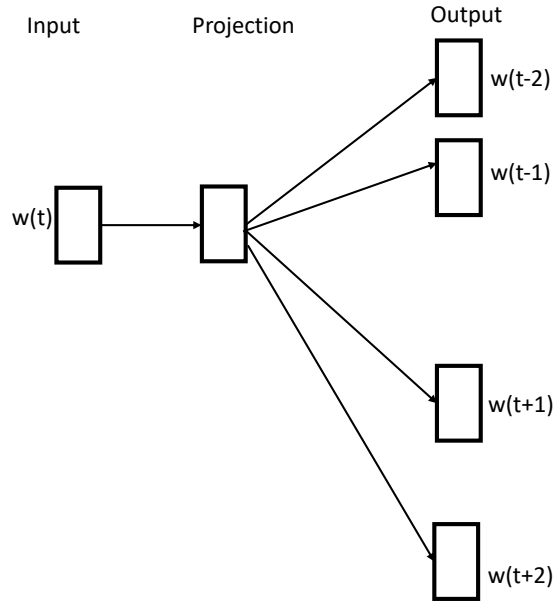


Figure 2.3: Neural Architecture for Skipgram Word2Vec

‘John’s school is five blocks away from his home’ are different. Such contextually embedded word vectors are typically useful for downstream tasks (e.g. sentiment classification and question answering), because use of contexts in such downstream tasks generally proves to be beneficial, e.g. the sense of the word ‘school’ in the two example sentences are substantially different.

Generally speaking, contextual word embedding methods employ a recurrent neural network based architecture (a language model) to predict the current word given its context. In contrast to Word2vec, where each unique word in the vocabulary has its own set of parameters irrespective of the context in which the word occurs, in context based embedding method, each occurrence of a word (as a part of a sentence) has its own set of parameters. ELMO (Peters et al., 2018) employs two-layer bidirectional language models implemented with stacked layers of character convolutions based LSTMs. After training the ELMO architecture on a large corpus of text, the vector representation of each word can be obtained by inputting the word to this trained model, which is similar in flavour to obtaining the vector representations of a new image from the output of a network (fully connected layer) pre-trained on a large collection of images (Krizhevsky et al., 2012).

As another approach to contextual embedding, (Devlin et al., 2019) proposed a bidirectional transformer (Vaswani et al., 2017) based approach, named BERT. The working principle of BERT involves training the parameters of a transformer based network on two pseudo-tasks (weakly supervised), namely: a) *Masked LM*, where words within a sentence are masked, and the training involves predicting these masked words; and b) *Next sentence prediction*, a binary classification task which given a pair of sentences predicts if one immediately follows the other or not. Both the above BERT pseudo-tasks are trained with both positive and randomly selected negative samples. Similar to ELMo, the vectors for a word are obtained by feeding in a word to the pre-trained architecture and obtaining the output. We use Word2vec (Mikolov et al., 2009) and BERT in Chapters 5 and 7 to address semantic similarity in proactive IR.

## 2.4 Summary

In this chapter, we have reviewed standard IR models including the VSM, BM25, LM, and relevance language model (RLM). An extension of relevance modeling technique has been used to address our RQ3 in Chapter 5. We also introduce the concept of word embedding which has been used extensively in our work in Chapter 5 and 7 to identify user activities that are related to similar information goal.

# Chapter 3

## A Framework for Proactive Information Retrieval

As introduced in Chapter 1, the objective of a proactive IR model is to help a user in accomplishing an overall information based goal or task. In this chapter, we introduce a generic framework for proactive IR, and propose a model for proactive query formulation.

### 3.1 Proactive IR Generic Framework

A user interacts with a digital application interface with a goal or a task in his mind. As a part of completing such a task, the user typically performs multiple activities. As explained in Chapter 1, the objective of a proactive IR model is to support the user in more efficiently or effectively undertaking their activities to help them achieve their goal. To do this a proactive IR system is required to track a user's activities, and then to automatically formulate queries on their user's behalf. Subsequently, these queries can then be used by the proactive IR system to search available resources to provide potentially useful information that may be helpful to the user. Figure 3.2 shows the general workflow of how a user's activity context is used in proactive IR to provide suggestions to the user. We describe each component

Activity Type	Activity Description	Timestamp
Click Document	Document Title: Just-in-time information retrieval agents	02/02/2021 14:40 pm.
Read	In this paper, just-in-time information retrieval agents(JITIR agents) are introduced. JITIR agents (pronounced “jitter agents”) are a class of software agents that proactively present information based on a person’s context in an easily accessible and nonintrusive manner.	02/02/2021 14:41 pm.
Write	The concept of proactive IR was first proposed in the work of Rhodes and Maes in 2000.	02/02/2021 14:43 pm.
Search	Proactive Search Research Paper	02/02/2021 14:44 pm.

Figure 3.1: Sample Activity Log of a User

of Figure 3.2 in detail in this section. First we turn our attention to describe the notion of a user activity in more detail.

In this research study, we focus on the type of user tasks which can be accomplished by a set of user interactions with a computing device (e.g. laptop or desktop). Hence we focus only on the desktop activities of a user. Examples of such user activities include reading a document in offline mode from a desktop folder or an online web document in the cloud, writing a document, clicking a document, or typing queries in a search interface. Figure 3.1 shows an example of a user’s desktop activity log. The task of the user in Figure 3.1 is to write a report on PIR. The user initially clicked a document and then started reading the document. After reading a few lines from the document, the user started writing the report on the topic of PIR. While writing the user also used a search interface to search a query related to PIR. We represent each activity in terms of a bag of words, as shown in the ‘Activity Description’ column of Figure 3.1. While logging a user’s desktop activities we create a new log each time a user switches application (e.g. from word editor to browser) or stops interacting with the system (i.e. the computing device goes in idle mode) or a maximum word limit is exceeded (e.g. the number of words used to

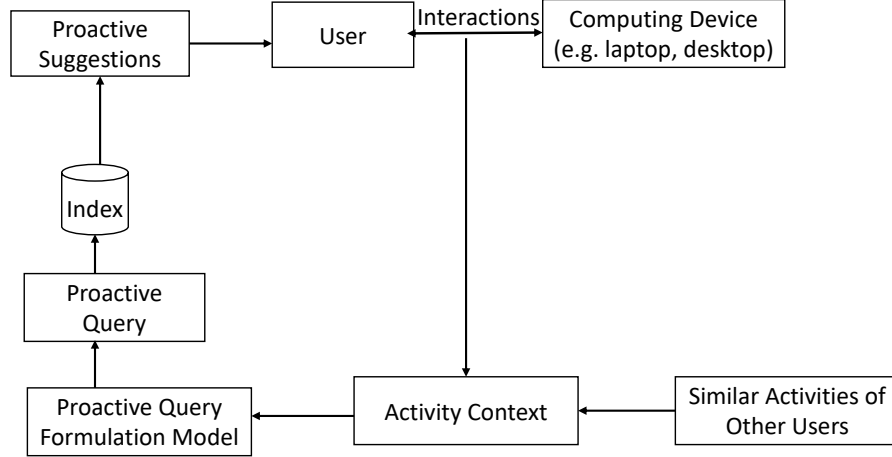


Figure 3.2: Generic Framework for proactive IR.

describe the activity exceeds a certain threshold).

### 3.1.1 The Generic Workflow of a Proactive IR Model

Given the activity context of a user, a proactive IR model should seek to anticipate the user’s potential information needs and formulate queries on the user’s behalf. Using a proactively formulated query, a proactive IR model uses a search system to retrieve a ranked list of potentially useful information resources from an index. Within our research scope, the index can be a collection of web pages (e.g. Clueweb12B<sup>1</sup>) or a collection of offline information dataset (e.g. TREC dataset). Since the index is comprised of a collection of documents, the possible forms for proactive suggestions retrieved from an index are documents, text snippets or document titles. The top  $k$  retrieved information items can be shown to the user as proactive suggestions.

Tracking a user’s activities and proactive suggestions is a continuous process. So it is necessary to determine the situations that should trigger proactive IR suggestions. This can most simply be taken to be after a fixed time interval, or after a fixed number of activities executed, or after switching a desktop application.

If we provide proactive suggestions after a fixed interval of time, then it may happen that the user’s information state may or may not have progressed significantly

<sup>1</sup><http://boston.lti.cs.cmu.edu/clueweb12/>



enough during that time interval. As an extreme case, if the user was completely idle during the time interval, then the proactive IR model will not have any new activity context which will lead to it providing proactive suggestions identical to the previous one.

Consequently, in our research scope, we propose to trigger proactive suggestions after a fixed number of activities or after switching a desktop application. If two consecutive sets of proactive suggestions are similar, then most of the latter suggestions are unlikely to be useful for a user. As a result of this, while during retrieving proactive suggestions at a particular timestamp, we drop any information resources that a user was already notified of previously.

After describing the workflow of a proactive IR model in Section 3.1.1, we now turn our attention to the potential sources of information that can be used in proactive query formulation which is an essential component in a proactive IR model.

### **3.1.2 Information Sources for Proactive IR Model**

The potential information sources for proactive IR can be broadly categorized into two types:

1. The user's own personal interaction history.
2. The interaction history of other users performing similar tasks or with similar interests to the current user

Each of the above mentioned types of source is described in the following subsections.

### **3.1.3 User's Personal History**

A user's personal history corresponds to the set of their activities that have been tracked by a proactive IR model up to the point of proactive suggestion. A user's personal history can be grouped into different sessions, where a session is defined as a sequence of consecutive activities where the time difference between a pair

consecutive activities is not higher than a particular threshold of time (e.g. 30 minutes). Essentially a user's personal history is comprised of a set of sessions from the user's past activity logs.

The importance of a user activity in anticipating a user's information need is inversely proportional to how far that user activity is from the most recent user activity (Kong et al., 2015). So we can broadly categorize a user's personal history into two sections:

- a) User's personal history from the current session.
- b) User's personal history from any session other than the current session.

The first category of activities indicates the user's current activity context, and the second category of activities indicates their previous contexts. Users have a multi-tasking nature, so from their past activities we need to try to identify those that are related to the user's current activity.

### **3.1.4 Similar Activity of Other Users**

Other users' activities related to the current user's current activities can also be a potential information source for proactive IR. There are a number of common tasks (e.g. planning for a vacation to a particular place, registering for a particular conference, etc.) which multiple users can be expected to perform. For common tasks, similar activities of other users can be used to potentially suggest useful information to the current user. For example a user is planning to attend the SIGIR 2022 conference. The different related activities for the task of arranging to attend the conference might include SIGIR 2022 registration, investigation of places to visit at the conference site, booking flights and accommodation. If we observe that a user is currently performing registration for the SIGIR 2022 conference (e.g. doing online registration, saving registration receipt in local desktop folder), then we can mine similar SIGIR planning related activities from other users' past activities to help the current user in completing his task. Examples of other users' past

$a_i$	$i^{th}$ Activity
$H$	Activity History of All Users
$\hat{q}_t$	Proactive Query at tim $t$
$L(\hat{q}_t)$	Ranked list of information Sources for $q_t$
$C_t$	Recent activity context of a user at time $t$
$H_u$	Activity History of a User
$H_o$	Activity History of other Users
$\phi$	Function for Computing ranked list of proactive suggestions

Table 3.1: Notations For Describing Proactive IR Model

activities in this scenario might include SIGIR registration, checking accepted paper list, bookmarking interesting talks, tutorials in SIGIR 2022, finding places to visit near SIGIR location.

Based on the identified information sources, we now describe our proactive IR model.

## 3.2 Anatomy of a Proactive IR Model

We start this section by first introducing the notations used to describe our proactive IR model. Table 3.1, shows the variables used in the definition of our proactive IR model. Each user activity  $a_i$  has a bag of words representation. The activity history  $H$  is comprised of the user’s own activity history (i.e.  $H_u$ ) and similar activity history to other users ( $H_o$ ). Hence we can say that  $H$  is union of both  $H_u$  and  $H_o$ .

In Table 3.1, the proactive ranked list of information sources  $L_{q_t}$  is obtained by choosing the top  $k$  documents from a ranked list of documents retrieved using proactive query  $q_t$  and an IR model (e.g. BM25 (Robertson and Zaragoza, 2009), LM (Ponte and Croft, 1998), RLM (Lavrenko and Croft, 2001)).

Given the sequence of activities, the first step of a proactive IR model is to formulate a query. To formulate a query proactively we first need to understand the importance of different words with respect to possible information goals for a user at a particular timestamp. In practice, we propose to realise this by estimating a distribution  $P(w|\theta)$ , where  $\theta$  is a relevance set at a particular time instant, addressing

the possible topics, along which the information seeking goal of the user may evolve. We construct a weighted query  $q_t$  by choosing the top  $n$  terms from  $P(w|\theta_R)$ . As explained in Section 3.1.2,  $\theta_R$  is generated from both: a) the set of recent activities  $C_t$ , b) activity history of Users  $H$ .

We describe how  $P(w|\theta_R)$  is computed from  $C_t$  and  $H$  in more detail in the next Section.

### 3.2.1 Proactive Query Formulation Model

As discussed in Section 3.2, we need to find a weighted distribution of words (i.e.  $P(w|\theta_R)$ ) to formulate a proactive query. To compute  $P(w|\theta_R)$  we first need to estimate the relevance distribution  $\theta_R$ . Recall from Chapter 2 that the RLM (Lavrenko and Croft, 2001) uses a relevance set similar to what we need here to compute the co-occurrence probability of words with a user typed query to estimate the importance of a word in expressing a user’s information need. The relevance language model (RLM) proposed in (Lavrenko and Croft, 2001) uses the top retrieved documents for a query to estimate the relevance set. We propose a modified version of (Lavrenko and Croft, 2001) to generate  $\theta_R$ , where the different information sources used to generate  $\theta_R$  are a user’s recent activity context  $C_t$ , and similar previous activities of other users (i.e.  $H(a_t)$ ). Our proposed relevance model is shown in Figure 3.3.

The rationale for making the construction of the induced subset ( $H(a_t)$ ) depends only on the current activity, is that subsequent information needs are most likely to depend only from the current one (Feild and Allan, 2013). Formally speaking,

$$\theta_R|a_t \sim B_t = C_t \cup H(a_t), \quad (3.1)$$

where  $B_t$  is used to denote the background information of the context from the current session and the past activities from the past activity log. In the following subsection we describe how we find similar activities from the past activity log ( $H$ ).

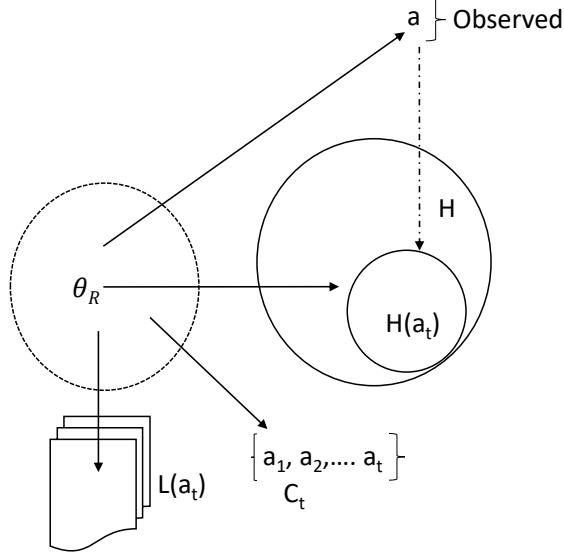


Figure 3.3: Generative model for proactive Query Formulation.

### Selecting Candidate Activities from History

While interacting with a digital interface a user is often engaged in multi-tasking (Lucchese et al., 2013). So it is not the case that a sequence of user activities will always be related to the same information goal. We select activities related to a user's most recent activity from  $H$ . Assuming that there exists some notion of a similarity function between a pair of activities,  $\sigma(a_i, a_j)$ , we select the set of candidate activities as the  $k$  top-most similar activities to the current activity  $a_t$ , i.e.,

$$H(a_t) = \cup\{a \in H : \forall r \in H - H(a_t) \sigma(a_t, q) > \sigma(a_t, r)\}, |H(a_t)| = k, \quad (3.2)$$

where  $k$  is a parameter determining how many related activities from the set  $H$  we should consider. Estimating  $H(a_t)$ , we will now discuss the computation of  $P(w|\theta_r)$  which is the core component of our proposed relevance model.

### Estimating the Relevance Model

Recall from Chapter 2 that the work in (Levine et al., 2017) proposed a time-decaying session-relevance model, where the weight of a previous query in estimating

relevance set was inversely proportional to its timestamp difference with the current query. Similar to (Levine et al., 2017) we also assume that it is more likely that an activity executed earlier will have less importance in estimating a user’s current information need. Given the observed current activity  $a_t$ , we use the formula shown in Equation 3.3 to compute  $P(w|\theta_R, a_t)$ .

$$P(w|\theta_R, a_t) = \prod_{t \in a_t} \sum_{a \in B_t} P(t|a)P(w|a)e^{-\delta(a_t, a)^2}. \quad (3.3)$$

Intuitively speaking, the probability of a term contributing to the topic of a potential proactive query in Equation 3.3 is high if it frequently co-occurs with a term appearing in the most recent activity,  $a_t$ .

In Equation 3.3,  $\delta(a_t, a)$  denotes the timestamp difference between an activity  $a_i$  in the current session,  $C_t$ , and the current activity,  $a_t$ . If we only use the co-occurrence probability of a word ( $w$ ) within an activity ( $a_i$ ) to estimate  $P(w|a_i)$ , then for all the scenarios where a word  $w$  does not appear in  $a_i$  at all, the value of  $P(w|a_i)$  will be equal to zero. So we use the standard notion of the Jelinek-Mercer (Jelinek and Mercer, 1980) method to smooth the probability of a word being sampled from an activity  $a_i$  as shown in Equation 3.4.

$$P(w|a_i) = \lambda \frac{f(w, a_i)}{|a_i|} + (1 - \lambda) \frac{c(w)}{S} \quad (3.4)$$

In Equation 3.4,  $f$  is the frequency of a term  $w$  in activity  $a_i$ ,  $c(w)$  is the collection frequency of  $w$ , and  $S = \sum_{w \in C_t \cup H(a_t)} c(w)$  is the collection size. Next, we define the temporal difference factor between activities as shown in Equation 3.5.

$$\delta(a_t, a) = \begin{cases} c & a \in H(a_t), c \geq (t - t') \\ t - t' & \exists t' : a = a_{t'} \in C(t). \end{cases} \quad (3.5)$$

The temporal difference (i.e.  $\delta(a_t, a)$ ) is used to model the anticipated importance of an activity in affecting future information needs within a session. The more recent

an activity is within a session, the more likely it is that it will have more importance in predicting a user’s future information need (Kong et al., 2015). Note that the function in Equation 3.5 ignores the temporal factor if the activity  $a_i$  is a part of the candidate set of activities extracted from the set  $H$  (previous queries from a log) because the timestamps of these activities should not have any direct effect on the likelihood of formulation of a statement of information need.

### 3.2.2 Generalization beyond Term Matching

In our query formulation model, we formulate a proactive query based on a user’s activity context. It may happen that there is no word overlap between a previous activity and the most recent activity of the user. This does not necessarily mean that the previous activity is not related to the user’s current task. However, the query formulation model shown in Equation 3.3 can not capture the task semantics between a pair of activities beyond term matching. As a result of this, the proactive suggestions retrieved using the proactive query, constructed using Equation 3.3, may not be effective for supporting the current task of the user. To address this limitation, we incorporate the notion of embedding based similarities in Equation 3.3, similar to the proposition of (Roy et al., 2016), which also incorporated embedding based similarity in the relevance language model (Lavrenko and Croft, 2001) to improve the effectiveness of an IR model.

Embedding based similarity goes beyond term matching, and hence can compute semantical similarity between a pair of activities even if they have little or no word overlap. More precisely, we replace the notion of the probability of sampling a word  $w$  from an activity  $a$ , as the likelihood of sampling  $w$  from  $a$  weighted by the cosine similarity between the embedded vector representations of  $w$  and  $a$ , as shown in Equation 3.6.

$$P_e(w|a) = P(w|a) \frac{\mathbf{w} \cdot \mathbf{a}}{\|\mathbf{w}\| \|\mathbf{a}\|}, \quad (3.6)$$

In Equation 3.6,  $P(w|a)$  is defined as per Equation 3.4,  $\mathbf{w}$  represents the vector of

a word  $w$  and  $\mathbf{q}$  represents that of a query  $a$ .

With these modified sampling probabilities,  $P_e(w|a)$ 's, that take into account the semantic similarities between a word  $w$  and a query  $a$ , we modify Equation 3.3 to the form shown in Equation 3.7.

$$P(w|\theta_R, a_t) = \prod_{t \in a_t} \sum_{a \in B_t} P(t|a) P(w|a) e^{-\delta(a_t, a)^2} \frac{\mathbf{a}_t \cdot \mathbf{a}}{|\mathbf{a}| |\mathbf{a}_t|}. \quad (3.7)$$

Intuitively speaking, Equation 3.7 considers two types of similarities, namely, similarity between a pair of queries and those between an individual term and a query. Assuming that task-specific semantics are captured with the help of a representation learning model (to be discussed in Chapter 5), terms that favorably contribute to the topic of a potential next query are those that:

- i) Occur in a query that is semantically similar to the most recent activity,  $a_t$ .
- ii) Frequently co-occur with terms appearing in  $a_t$ .
- iii) are semantically similar to the terms of  $a_t$ .

### 3.3 Concluding Remarks

This chapter described a generic framework for proactive IR in Section 3.1. We also proposed a query formulation model for proactive IR in this chapter. The proactive query formulation framework proposed here will be used in two different application scenarios (i.e. single Stage and multi Stage) in Chapter 6 and Chapter 7 respectively. In the next chapter we develop an evaluation framework for proactive IR.



# Chapter 4

## Evaluation of Proactive Information Retrieval

In this chapter, we address our first research question ‘How can we quantitatively evaluate the effectiveness of proactive IR models?’ Existing work in proactive IR has collected user feedback to evaluate a proactive IR model. However, the aim in our research is to create and use a reproducible evaluation framework. Here we first outline the standard evaluation metrics for IR and describe the challenges in applying these to proactive IR. We then describe how we propose to extend the use of these metrics for evaluation of our investigation of proactive IR.

### 4.1 Evaluation Metrics in Traditional IR

The objective of a traditional IR model is to satisfy a user’s information need. Generally speaking, a user is usually considered to be satisfied if he observes relevant documents corresponding to his information need towards the top ranks of a list of documents retrieved in response to their query submitted to an IR system. Another criteria for user satisfaction is the number of relevant documents shown to the user.

To evaluate the quality of a ranked list one should compute the ranks or positions of the relevant documents within the retrieved list. In order to do so automatically

it has to be known (ideally speaking, subjectively) which documents are relevant to a given query. However, a subjective knowledge of relevance requires explicitly asking the user to provide feedback (e.g. click a check-box) on whether a document is relevant or not. Collecting personalized (subjective) relevance judgments makes it difficult to establish benchmarks for comparing across different retrieval models from an objective standpoint.

To facilitate laboratory-based IR evaluation from an objective standpoint, existing IR research employs a *pooling mechanism* (Voorhees, 2003), which involves executing a number of different retrieval approaches, combining the top- $k$  results (formally known as depth- $k$  pooling) and conducting manual assessments (of objective relevance) on the contexts of the pool.

This set of manually judged documents for each query constitutes a *groundtruth* or *reference* set of objectively relevant documents (Voorhees, 2003) that are judged (assessed by humans) to be relevant for that query. The judged documents can be associated with a binary or a graded level of relevance (Järvelin, 2010; Robertson et al., 2010), each with its own set of prescribed methodologies for evaluation.

Before reviewing the rank-based IR metrics (with binary or graded judgments), we first examine the concepts of precision and recall in IR evaluation in the next section.

#### 4.1.1 Set-based Evaluation Metrics

The information need of a user can typically be satisfied in two different ways, each with its own advantages and disadvantages, both of which an IR evaluation metric should consider. We present the two different situations with the following two illustrative examples.

1. An IR system presents a small number of relevant documents (out of the total number of relevant documents available in the collection, e.g. only 4 out of 32) to the user towards the very top of the search result list. For example, the

first page of a paginated result list (comprising, say, 8 documents) the user could be presented with 4 relevant documents (Kelly and Azzopardi, 2015a). However, no further relevant documents are retrieved in the remaining of the search engine result pages (SERPs).

2. An IR system is able to retrieve a large number of relevant documents, e.g. say 25 out of 32. However, these documents are retrieved from the fifth SERP onwards.

The first situation is an example of a *precision* oriented system (or a high precision only system), whereas the latter exemplifies a *recall* oriented one. Precision-oriented systems are useful in situations when a user is typically satisfied with a small number of relevant documents and is not prepared (or does not need) to read the other relevant documents, e.g. only one or two news articles on a recent topic would be adequate for a user. However, in other situations, such as patent prior art search or legal search (Ganguly et al., 2011b), it is crucial to locate as many relevant items available as possible.

We now define the concepts of precision and recall formally. Both of these are *set-based measures* extensively used to evaluate the quality of predicted outputs of IR systems against a groundtruth set of relevant items.

**Precision:** It is defined as the number of correct predictions or true positives out of a total number of predicted positives. To compute precision in the context of IR, one needs to cut-off the ranked list of retrieved documents at a particular position (say  $k$ ) and view it as a set (rather than an ordered list). Precision is then given by the *number of relevant documents* within this set,  $L_k = \{D_1, \dots, D_k\}$ , of  $k$  documents (analogous to the number of predicted positives). This metric is formally known as the *precision at top- $k$*  and is denoted symbolically as  $P@k$ .

$$P@k = \frac{\sum_{i=1}^k \mathbb{I}[R(D_i) = 1]}{k}, \quad (4.1)$$

where  $R(D_i)$  denotes the binary relevance (1/0) of the  $i^{th}$  document of the set of

top- $k$  documents, and  $\mathbb{I}[X]$  denotes the indicator variable which is 1 if the property  $X$  is true.

**Recall:** Recall is defined as the number of correct predictions or true positives out of the total number of true positives. On a similar note, the general definition of recall also requires a cut-off of the ranked list of retrieved documents at a particular position (say  $k$ ), and then to compute the *number of relevant documents* within this set of  $k$  documents (analogous to the number of predicted positives) and (*different to precision*) normalize it with the *total number of relevant documents in the collection*. This metric is formally known as the *recall at top- $k$*  and is denoted symbolically as  $R@k$ . The formula for  $R@k$  is shown in Equation 4.2.

$$R@k = \frac{\sum_{i=1}^k \mathbb{I}[R(D_i) = 1]}{R}, \quad (4.2)$$

where  $R$  denotes the total number of relevant documents in the collection (for the current query),  $R(D_i)$  denotes the binary relevance (1/0) of the  $i^{th}$  document of the set of top- $k$  documents, and  $\mathbb{I}[X]$  denotes the indicator variable which is 1 if the property  $X$  is true.

### 4.1.2 Rank-based IR Measures Using the Concepts of Precision and Recall

Cutting off the ranked list at an arbitrary position poses a number of problems. First the ideal cut-off point is not clear. Second, a set-based measure ignores the ranks of the retrieved items.

**Example 4.1.1.** *Relevance of documents retrieved by two example systems.*

*System A:*  $\{1, 1, 0, 0, 0\}$ ,  $P@5(A) = 2/5$

*System B:*  $\{0, 0, 0, 1, 1\}$ ,  $P@5(B) = 2/5$

In Example 4.1.1 the  $P@5$  values of both systems A and B are  $2/5$ , which is unfair because from a user satisfaction point of view, system A is more preferable

than B because A saves the user effort in reading to scroll down the ranked list to locate the relevant documents (retrieved by B at positions 4 and 5).

These issues of arbitrary rank cutoffs and the rank agnostic nature of set-based metrics, are alleviated by the metric - Average Precision (AP) which is defined as follows.

**Average Precision (AP):** AP is the aggregated precision values normalized by the total number of relevant documents for a query, where instead of using arbitrary rank-cutoff points, the precision values are computed using *recall points* (a position where a relevant document is retrieved).

An example of recall points for the system A in Example 4.1.1 are the positions 1 and 2 (because these are the ranks at which relevant documents were retrieved by A). To take into account recall alongside the aggregated precision measures, the metric is normalized by the total number of relevant documents in the collection. Formally speaking, average precision is defined as shown in Equation 4.3.

$$\text{AP} = \frac{1}{R} \sum_{i:R(D_i)=1} P@i = \frac{1}{R} \sum_{i:R(D_i)=1} \frac{\sum_{j=1}^i \mathbb{I}[R(D_j = 1)]}{i} \quad (4.3)$$

Assuming that there are 5 relevant documents in total for the situation shown in Example 4.1.1, the AP value of system A is computed as

$$\text{AP}(A) = \frac{1}{5} \left( \frac{1}{1} + \frac{2}{2} \right) = \frac{2}{5},$$

whereas that of B is given by

$$\text{AP}(B) = \frac{1}{5} \left( \frac{1}{4} + \frac{2}{5} \right) = \frac{13}{100},$$

which, as it ideally should be, is less than that of A's.

AP values, aggregated over a number of benchmark queries, is referred to as mean AP (commonly known as MAP). This metric is the most well-known metric in the evaluation of the effectiveness of a set of benchmark queries. It has been used

in a large number of research papers and evaluation forums to compare the relative performance of different retrieval systems. The computation of MAP for a set of queries  $Q$  is given in Equation 4.4.

$$\text{MAP} = \frac{1}{|Q|} \sum_{q \in Q} \text{AP}(q) \quad (4.4)$$

### 4.1.3 Evaluation Measure of Non-Binary Relevance Assessments

Metrics such as precision and recall are conceptually well-defined only for binary relevance (i.e. whether a prediction is correct or not). However, the notion of relevance, for practical purposes, can be non-binary, e.g. some documents provide more useful information to a user on a particular topic than another document, in which case the former is an instance of relevant document whereas the latter is an instance of a partially relevant one.

A commonly used metric for evaluation using graded relevance is *NDCG* (Normalized Discounted Cumulative Gain) (Järvelin and Kekäläinen, 2002). The underlying principle of *NDCG* is to compute how *similar* a retrieved list is to the *ideal retrieved list* for a given query. Obviously, the ideal retrieved list is one which presents the documents in decreasing order of relevance. For instance, in Example 4.1.1, the ideal list is  $\{2, 1, 1, 1, 1\}$  (assuming one among the relevant documents is fully relevant, the others being partially relevant).

To compute *NDCG* up to rank  $p$  (i.e.  $NDCG_p$ ), one first computes *discounted cumulative gain* up to rank  $p$  (i.e.  $DCG_p$ ) and ideal discounted cumulative gain up to rank  $p$  (i.e.  $IDCG_p$ ).  $DCG_p$  is computed using the formula  $\sum_{i=1}^p \frac{rel_i}{\log(i+1)}$ , where  $rel_i$  determines the graded relevance for  $i^{th}$  retrieved document.  $IDCG_p$  is computed using the formula  $\sum_{i=1}^{|REL_p|} \frac{2^{rel_i-1}}{\log(i+1)}$ , where  $REL_p$  is the ordered set of top  $p$  documents based on the relevance score.  $NDCG_p$  is the ratio of  $DCG_p$  and  $IDCG_p$  as described in Equation 4.5.

$$NDCG_p = \frac{DCG_p}{IDCG_p} \quad (4.5)$$

## 4.2 Differences Between Standard IR and Proactive IR Evaluation Criteria

In contrast to standard IR, a proactive IR model tries to anticipate a user’s information needs from his current and recent activities (i.e. formulates queries on the user’s behalf), rather than addressing the user’s information need as actively expressed in a query. A proactive query is used to provide a user with potentially relevant documents that may help him in completing his current task activities, which may be single staged or multi-staged as described in Chapter 1. The objective in proactive IR is to help in the overall task of the user, where a task may involve one or more related queries, whereas in standard IR, we focus on satisfying a user’s information need on an individual query basis.

Since the objective of a PIR model is to support the user in his overall task, the effectiveness of a PIR model should be measured over all the proactive suggestions made during the completion of a task by a user rather than for individual queries. It can though be difficult to automatically identify task boundaries. For example, these can span more than one session and consecutive sessions may not always be related to the same information goal. However, existing work shows that it is likely that a user will be engaged in a single task within an individual session (Wang et al., 2013; Tran et al., 2016). A session typically consists of a sequence of activities where the time gap between a pair of consecutive activities is not more than a particular threshold of time (e.g. 26 minutes (Lucchese et al., 2013)). Existing IR metrics such as session-NDCG (Kanoulas et al., 2011) measure the effectiveness of an IR model in a multi-query session. However, they use a single ranked list for a session, where the list is constructed by concatenating the top  $k$  documents corresponding to each query in the session. However in PIR, the objective is to always provide new documents

in each of the proactive suggestions within a session. Hence the evaluation metric for PIR should be averaged over all the suggestions within a session rather than considering them as a single list.

In the following section we describe the desirable characteristics of an evaluation metric for proactive IR.

### 4.3 Desirable Characteristics of Proactive IR Evaluation Metric

Before describing the desirable characteristics for our proposed mechanism, we recapitulate our assumptions of the general workflow of a proactive IR model, (c.f. Section 3.1) as defined in the scope of our study. Firstly, the workflow involves an *intermediate step of query construction* (or query formulation), within a given activity context. System estimated queries are then used to retrieve a list of documents to be presented as proactive suggestions to the user.

In our research, a proactive IR model provides proactive suggestions after a fixed number of user activities, as described in Section 3.1. While providing proactive suggestions a user may or may not have an explicit information need. If the user has an explicit information need at the point of a proactive suggestion, then we can compare the query formulated by a human user to express this information need at that point with the proactive query created by the PIR model.

The effectiveness of a user formulated query depends on the search expertise of the user and also the complexity of the overall task in which the user is engaged. It may no be the case that a user created query is the ideal representation of their information need. However, in our evaluation framework, we do not consider a user’s search expertise or the complexity of the task as a parameter. We thus consider the user formulated query to be the reference point that the query estimation process should ideally replicate.

The evaluation framework also assumes that at each point of proactive suggestion



there will be a *reference* set. The *reference* set at the  $i^{th}$  proactive suggestion (i.e.  $R_i$ ) is the collection of documents that may be useful for the user in completing his overall task. As described in Section 4.2, a difference from standard IR is that we assume that the relevance of a document in the reference set pertains to the entire session (or task) rather than to a single query. In Section 4.5, we describe in detail how we compute the *reference* set for proactive IR evaluation.

After describing the key elements of the workflow of a proactive IR system, we now take a closer look into the desirable characteristics of a proactive IR evaluation mechanism from a general stand point.

**$C_1$ : Similarities between the actual query and the predicted query by a PIR model:** An ideal PIR model should be able to predict the explicit information need that may arise in a user’s mind while engaged in a task. As described earlier, in this section we consider a user entered query as the reference point for a user’s information need representation. Hence the similarity between the predicted query and the user entered query should be one of the characteristics of our proposed evaluation metric.

**$C_2$ : Similarities between predicted and reference set of documents:** The output of a proactive IR model is a ranked list of suggested documents. The second desirable characteristic of a evaluation mechanism for proactive IR is that it should focus on measuring the effectiveness of the output of the model ( i.e. the ranked list of document suggestions). As described in Section 4.3, the reference set contains the groundtruth documents at the point of each proactive suggestion. Thus, the evaluation metric should compare between the predicted list of recommended documents (i.e. the output of the proactive IR model) and the reference set of documents, to measure the effectiveness of the proactive IR model.

**$C_3$ : Average of evaluation metric across proactive suggestions:** The third characteristic of an evaluation metric for proactive IR is that rewards obtained for the predicted list should be accumulated for each proactive suggestion within a session that follows from the point within the session at which the model becomes

proactive. This characteristic ensures that a proactive model is evaluated across a whole session in which the user was involved in a task. The performance of a proactive IR model on a single task can not determine its effectiveness alone. Hence we should average the performance of a proactive IR model across multiple tasks to estimate the overall performance, rather than across multiple individual queries, as in the case of standard IR.

**$C_4$ : Rewards should preferentially be weighted on context length of activities:** The evaluation metric should be able to favour systems that start recommending early (i.e. capable of addressing the cold-start problem). This aggressive behaviour of starting early is desirable from a user point of view because it helps to reduce the potential user effort of actually constructing queries during the session. Moreover the earlier useful information is proactively provided, the easier the task will be for the user or less effort will be required from them.

On the other hand, a more conservative approach of waiting longer before commencing recommendation in order to accumulate adequate session context (so as to help to obtain higher quality recommendations) means that a user has to undertake more activities (e.g. more user constructed queries, more documents read by the user, etc.). This in turn means that the very objective of reducing user effort would be compromised since they will have completed much of the overall task before any information is provided to them proactively. So the rewards at a particular point in proactive suggestion should be inversely proportional to the length of the context of previous user activities.

## 4.4 Proposed Evaluation Metric

Based on the characteristics discussed above we propose two broad categories of evaluation metric: one for evaluating the quality of proactive queries, and the other for evaluating proactive suggestions within task sessions. The reason for choosing two different categories of metric is as follows. Firstly, there may be scenarios

where only one of the groundtruth information categories is available (e.g. query or reference set). For example, we may be aware of the query that a user entered given an activity context, but we may not know the set of relevant documents which may be of use to the user at that time. Secondly, it may not always be the case that better queries provide better proactive suggestions. There may be situations where the proactive queries predicted by two different PIR models are not similar, but using which the ranked list of suggestions provided to the user are similar. One reason for this may be the absence of documents related to one of the proactive queries in the index. Since a user can only see the proactive suggestions, we should not focus only on evaluating the proactive queries. Similarly if we only focus on the evaluation of proactive suggestions, there may be scenarios where the anticipated information need is similar to a user’s actual information need, but the proactive suggestions provided to the user are not useful due to the retrieval model or index from which the suggestions are retrieved. As a result of this we propose to evaluate a PIR model based on both proactive queries and proactive suggestions.

The details of our proposed evaluation metrics follow below.

#### 4.4.1 Evaluating Proactive Queries

In Section 3.2.2, we described how we estimate the likelihood of a term for query formulation given a user’s activity context. To compute a query, we first sort all the terms in decreasing order of estimated likelihoods computed using Equation 3.7. Following this, we choose the top  $k$  terms starting from the each position in the sorted list to generate a list of queries. The first query generated is the most likely query given a user’s activity context. The further we go down the generated query list, the lower the number of overlapping words between the query and the top ranked generated query.

To satisfy the characteristics discussed in characteristic Section 4.3, we use mean reciprocal rank (MRR) to find the rank of the user created reference query in the generated query list. Since the queries generated in the list have word overlap, the

higher the value of MRR, the higher is the likelihood of word overlap between the user created query and the query estimated by the proactive IR model.

#### 4.4.2 Evaluating the Ranked List of Potentially Useful information Sources

We first describe the notations used for the proposed evaluation metric and then we move towards describing the proposed evaluation metric.

##### Notation

We use the notation shown in Table 4.1 to describe the evaluation metric for the ranked list of potentially useful information sources. We denote the evaluation metric as  $P(\pi, M, n)$ , which is parameterized by the  $\pi$ -index (i.e. the point from which an IR model becomes proactive during a task session), the number of recommended documents shown as proactive suggestions to the user  $M$  and the total number of proactive suggestions within a session  $n$ .

#Number of proactive recommendation in a session	$n$
#Number of activities in a session	$N$
Proactivity starting point ( $\pi$ -index)	$\pi$
#Documents recommended by a Proactive Model	$M$
Reward function for a single proactive recommendation	$r(k, M)$
Groundtruth of documents at $k^{th}$ activity	$R_k$

Table 4.1: Notation for proactive evaluation metric.

Recall from Section 4.3 that along with measuring the quality of proactively formulated queries, we also need to evaluate the predicted ranked lists of suggestions to measure the effectiveness of the proactive IR model. Here we focus on how similar the predicted list of documents is to a groundtruth list of documents (i.e. the reference set explained in Section 4.3). We call this measure a reward function (i.e.  $r(k, M)$ ) since it computes how much reward should be given to a proactive IR model based on the quality of the proactive suggestion list. In the  $r(k, M)$  function,  $k$  denotes the  $k^{th}$  proactive suggestion within a session.

Another parameter in the reward function is the number of top ranked documents,  $M$ , returned by the proactive IR model based on which the similarity between the predicted and the groundtruth list is computed. A user may not wish to scroll through a long suggestion list. Hence we compute our metric on the basis of the top  $M$  documents (e.g.  $M$  can be 3, 5 or 10) from the retrieved list obtained from the query estimated by the IR proactive model. The values of individual reward functions for each proactive query are presented in a general form in Equation 4.6, where  $\phi$  denotes a function to measure the similarity between the predicted list and the reference set  $R_k$  at the  $k^{th}$  proactive suggestion.

$$r(k, M) = \phi(L(a_{k+1}), R_k). \quad (4.6)$$

This definition of the reward function is consistent with characteristic  $C_2$  defined in Section 4.3. The reward functions are accumulated over a session as shown in Equation 4.7.

$$P(\pi, M, n) = \frac{1}{n} \sum_{k=\pi}^{\pi+n} \frac{r(k, M)}{k}. \quad (4.7)$$

As per characteristic  $C_3$  defined in Section 4.3, the total reward function in Equation 4.7 is averaged over the total number of proactive suggestions within a session (i.e.  $n$ ). As per the characteristic  $C_4$ , rewards should preferentially be weighted on the context length of activities. Hence the metric in Equation 4.7 discounts rewards at a point within a session by a factor of  $\frac{1}{k}$ , where  $k$  is the length of the context from the beginning to that point. If a system starts predicting early, the value of  $\frac{1}{k}$  will be low and it will favour the system. But as described in  $C_4$ , if the suggestions are of low quality then the value of  $r(k, M)$  will be low and the overall performance of the system will not be good.

**Variations of Reward Function.** The general evaluation metric  $P(\pi, M, n)$  defined above can be computed with multiple reward functions. Some suitable reward functions are as follows.

1. **MRR:** The mean reciprocal rank of the first relevant document in a ranked

list. MRR is a useful metric here since it shows how effectively we can provide potentially relevant information to a user through proactive suggestion. The higher the MRR value, the less effort is required by the user to find a relevant document in the suggested list.

2. **P@k:** Since we will show only the top  $k$  documents retrieved using the proactive query, the precision value up to rank  $k$  indicates how many potentially relevant documents we could show to the user.
3. **Ranked Correlation Coefficient:** If we have a relevance order in the reference list, we can also use a ranked correlation coefficient (e.g. Kendall's tau (Kendall, 1938) to compute the similarity between the ranked list of proactive suggestions and the reference list. This metric will show how much the ranking of the documents in proactive suggestion correlates with reference list.

**Cumulative Recall:** Along with  $P(\pi, M, n)$ , we also use another recall based metric to measure the effectiveness of a proactive IR model. This measures the proportion of the total number of unique relevant documents corresponding to a user's task that are retrieved across all proactive suggestions during a session. Rather than penalizing the reward function at each suggestion, here we simply add the reward function across each session. We show the mathematical formulation of our Cumulative Recall metric in Equation 4.8, where  $R_i$  is the number of new relevant documents appearing at the  $i^{th}$  proactive suggestion and  $N$  is the total number of actual relevant documents corresponding to the task. Cumulative Recall measures how useful a proactive model is with respect to locating information of potential value to the user in carrying out their task.

$$CumulativeRecall = \frac{1}{|N|} \sum_{i=1}^n (R_i) \quad (4.8)$$

### 4.4.3 Analysis for Variations in $\pi$ -Index

One of the important features of a proactive IR model is the point from which it starts to make proactive suggestions (i.e.  $\pi$ ). In this section, we analytically show that it is possible for our proposed metric  $P(\pi, M, n)$  (Equation 4.7) to score a system B higher than A, even if A starts recommending earlier than B. This analysis ensures fairness between two proactive IR models with different  $\pi$ -index values. For simplicity, we base our analysis on the MRR-based reward function. Let two systems be  $A(\pi_A, M, n)$  and  $B(\pi_B, M, n)$ , where  $\pi_A < \pi_B$ . Let the reward function of system A for activity  $a_k$  be  $m_k^A$  and that of B for the same activity be  $m_k^B$ . If system B outperforms A, then

$$P_B(\pi_B, M, n) > P_A(\pi_A, M, n),$$

which on substitution from Equation 4.7 gives Equation 4.9.

$$\sum_{k=\pi_B}^n \frac{r_B(k, M)}{k} > \sum_{k=\pi_A}^n \frac{r_A(k, M)}{k} \geq 0. \quad (4.9)$$

As described in Section 4.4.2,  $r(k, M)$  can have different types of variants. Without loss of generality, we choose MRR from among the different variants introduced in Section 4.4.2. After substituting MRR values, Equation 4.9 is transformed into the form shown in Equation 4.10.

$$\sum_{k=\pi_B}^n \frac{1}{k} \left( \frac{1}{m_k^B} - \frac{1}{m_k^A} \right) > \sum_{k=\pi_A}^{\pi_B-1} \frac{1}{k m_k^A} \geq 0. \quad (4.10)$$

In Equation 4.9,  $m_i$  denotes the rank of the first relevant document in the  $i^{th}$  ranked list. Now, with reference to Equation 4.10, we consider two scenarios where the desirable ranking order between two systems A and B are different.

**Case-1.** This corresponds to the scenario where System A performs better than B from the point B starts prediction, so ideally the ranking order between the two systems should be  $A > B$ .

To see if this could happen, we substitute  $m_k^A \leq m_k^B, \forall k = \pi_B, \dots, n$  and obtain

$$0 \leq \sum_{k=\pi_A}^{\pi_B-1} \frac{1}{km_k^A} \leq 0,$$

which shows that  $P_B(\pi_B, M, n) > P_A(\pi_A, M, n)$  is a contradiction. In other words, the metric indeed scores system A better. This is desirable because B's predictions were late and not better than A's.

**Case-2.** This case corresponds to the situation when System A performed well before B's recommendations after which B performed well and A poorly. The ideal rank in this case is  $A < B$ . To see what could happen in this scenario, let  $m_k^A = 1 \forall k = \pi_A, \dots, \pi_B - 1$  (A's recommendation was initially better than B's),  $m_k^A = \epsilon \forall k = \pi_B, \dots, n$  (A fails to recommend well after B starts), and  $m_k^B = 1 \forall k = \pi_B, \dots, n$  (B recommends better than A throughout). Substituting these values in Equation 4.10, and using the identity  $\log(n) \approx \sum_{i=1}^n \frac{1}{i}$  we get the condition described in Equation 4.11.

$$\begin{aligned} \sum_{k=\pi_B}^n \frac{1}{k} (1 - \epsilon) > \log(\pi_B - 1) &\Rightarrow (1 - \epsilon) \log\left(\frac{n}{\pi_B}\right) > \log(\pi_B - 1) \\ &\Rightarrow \epsilon < \log\left(\frac{n}{\pi_B(\pi_B - 1)}\right) \end{aligned} \quad (4.11)$$

Equation 4.11 suggests an upper bound on A's effectiveness to satisfy the condition that B is scored better than A in terms of  $P(\pi, k, M)$ . Equation 4.11 ensures that the metric is fair because it can score B better than A, if B (from its inception point which is later than A's) performs significantly better than A, otherwise A is favoured by  $P(\pi, k, M)$ .

## 4.5 Reference Set ( $R_k$ ) Computation

As shown in Equation 4.7, to compute the reward function one would need a reference set of documents  $R_k$  at each timestamp within a session. We described in



Chapter 1 that we propose proactive IR models for both single stage and multi-stage task scenarios. Since these scenarios are different, the process of computing  $R_K$  is also different for these scenarios. We now describe each of these scenarios and the corresponding  $R_k$  formulation.

#### 4.5.1 Reference Set Computation for Single Stage Task

When performing a single stage task, a user will relate to a single informational activity. We try to formulate one or more queries from a user’s recent and past desktop activities related to the user’s current task, and use these proactive queries to retrieve corresponding ranked lists of documents for the user.

As time progresses, we aim to enhance proactive queries with more contextual clues from the user’s activities as a session progresses, and to provide new relevant documents corresponding to anticipated information needs of the user based on formulated queries and search operations.

We broadly categorize a user’s desktop activities into read and write activities. There is no publicly available dataset of desktop activities where a real user is involved in a task which has a single information need associated with it. Hence for our examination of single stage proactive IR in Chapter 6, we use a simulation setup to generate a dataset of user reading and writing activities. The simulation is based on the scenario where a user is writing an article on a given topic or a user is interested in a particular topic and is reading documents related to that topic. We use the TREC Novelty track dataset for the simulation. The TREC Novelty track (Harman, 2002) is a collection of task topics where for each topic we have user judged relevant documents, and also details of manually labelled relevant sentences within each relevant document.

The simulation process starts with a user having a prior knowledge about a topic. Different users can have different levels of initial knowledge. In the current session, the user is engaged in desktop activities relating to a topic. The proactive IR model suggests documents to the user after a fixed numbers of activities. Recall from

Section 4.3, that the reference set is the set of ground truth documents against which we can measure the effectiveness of proactive suggestions for this session. So for the simulation setup for single stage tasks, the reference set (i.e.  $R$ ) consists of all the documents judged relevant corresponding to the task in which a user is engaged in the current session. The objective of a proactive IR model is to provide new relevant information sources with every suggestion. Hence after the  $i^{th}$  proactive suggestion, we remove the set of relevant documents retrieved upto the  $i^{th}$  suggestion from the set  $R$ . The updated  $R$  set acts as the reference set for the  $(i + 1)^{th}$  suggestion (i.e.  $R_{i+1}$ ).

#### 4.5.2 Reference Set Computation for Multi-Stage Task

For a multi-stage task, there are multiple sub-tasks within a task. The objective of a proactive IR system in this setting is to provide the user with relevant documents related to each sub-task as the user progresses through the task. The idea of proactive suggestion comes from exploiting similar multi-stage tasks of other users and also from the user’s own activities when executing similar sub-tasks in the past to support the task of current user. Again, there is no existing real life dataset corresponding to the information needs of multi-stage tasks of different users. Ideally we should install desktop activity tracking application on thousands of personal desktops, and collect the desktop activity logs of the users of these machines for a period of months and store this data using privacy preserving protocol (Tripathy and Pradhan, 2012; Pise and Uke, 2016). Developing an effective tracking application and deploying it on thousands of desktops would take considerable resource and an enormous amount of infrastructural support. Hence it was not feasible to create such a collection within this project. We do though have access to publicly available standard search query logs. Hence in our investigation of multi-stage proactive IR in Chapter 7, we limit our investigation of multi-staged tasks to multi-stage search tasks. One example of such a task is planning for a vacation. Since we consider search tasks for multi-stage information needs, the user activities we consider are

only queries typed by a user and the corresponding click logs. Since we do not have relevance judgements for queries in a query log, we assume that the top  $k$  retrieved documents corresponding to a user typed query  $q_i$  as relevant documents for the query, and we use these sets of documents as the reference set for each query. For an overall search task within a search session, the reference set is the union of all the top  $k$  retrieved documents for all the queries within that search session.

Since for a multi-stage task, we have a ranked list of documents retrieved for each query, we use another variation of the reward function along with the ones described in Section 4.4.2. We use Spearman correlation to compute the correlation between the predicted ranked list of documents and the ground truth ranked list of documents. Spearman correlation satisfies the characteristic  $C_2$  described in Section 4.3 and measures how similar the predicted ranked list of suggestions are with respect to reference set of documents.

## 4.6 Concluding Remarks

In this chapter we addressed our first research question which is about effective evaluation of PIR models. We first introduced the challenges of PIR evaluation by contrasting this with standard IR evaluation and its metrics. Then we proposed an evaluation framework to measure the effectiveness of proactive IR models. In Section 4.4.3 we showed how in different scenarios our proposed evaluation metric can effectively evaluate PIR models. The proposed evaluation metric is used in Chapter 6 and Chapter 7 to measure the effectiveness of proactive IR models in single stage and multi-stage task scenarios.

# Chapter 5

## Identifying Similar User Activities

In Chapter 3, we described a generic framework for proactive IR. Recall from Section 3.2.1 in Chapter 3, that to formulate a proactive query we need to find activities having similar overall information goals as the user’s current activity. In this chapter, we focus on methods for finding user activities related to similar overall information goals. More specifically, we examine our second research question which can broadly be stated as ‘*Can we group activities related to a similar task (i.e. overall information goal) together?*’. Generally speaking, a user’s task can be related to single or multiple sub-tasks (i.e. information goals), each of which can have nested information needs within it (Hassan Awadallah et al., 2014). An example of a task is planning to participate in a conference. Different sub-tasks within this task are selecting a flight, finding a hotel, making arrangements for local transport, finding the conference venue, finding good places to eat around the venue, finding local sight-seeing options after the conference, etc.

### 5.1 Introduction

Before delving into the details of grouping activities related to a task, we first introduce the concept of a ‘session’. We define a session as a set of activities where the time gap between a pair of consecutive activities is not more than a particular

threshold of time (e.g. 26 minutes (Lucchese et al., 2013)). Previous studies (Lucchese et al., 2013) have shown that users have a multi-tasking nature, so activities related to the same task may or may not be within the same session. Thus for the task of planning for a conference visit, sub-tasks such as selecting a flight and making arrangements for a hotel may occur in different sessions. We refer to the problem of automatically identifying the activities related to the same task as the *cross-session task extraction problem*, since the same task related activities can span across multiple sessions. Cross-session task extraction is a challenging problem, as can be seen from the following analysis. Firstly, note that it is likely that a session for a flight booking and one for local sightseeing around a conference venue may be far apart in time, as a result of which simple approaches of grouping activities based on the time of the search activities, e.g. (Lucchese et al., 2013), are not likely to yield satisfactory outcomes. Secondly, the term overlap between the activities of these two sessions is also likely to be low since they correspond to two different sub-tasks of the same task. This indicates that using lexical similarity for clustering cross-session activities into a single group is unlikely to be effective ( e.g. (Lucchese et al., 2013; Wang et al., 2013)).

As an illustrative example of term mismatch, consider the two queries ‘Eric Harris’, ‘Reb Vodka’ which are part of a user search activity from the AOL query log <sup>1</sup>(i.e. part of user search activity). Although these two queries do not share any common terms, they refer to the task of finding information on the Columbine high school massacre, the first query referring to the name of the first murderer, while the second one refers to their nickname.

To alleviate the identified problems in attempting to group activities by their timestamps or lexical similarities, we propose to embed activities in a task-based semantic space in a manner that will give similar activities in this space a high likelihood of pertaining to the same underlying task. Word embedding algorithms, such as ‘word2vec’ (Mikolov et al., 2013a), make use of lexical context in learning seman-

---

<sup>1</sup>[https://archive.org/download/AOL\\\_search\\\_data\\\_leak\\\_2006](https://archive.org/download/AOL\_search\_data\_leak\_2006)

tic representations of words. Moreover standard word embedding approaches do not take into account task aware context of words. So we propose to *transform* these representations into a task-oriented semantic space with the objective of making two words that are likely to be a part of the same task closer to each other.

To learn the transformation function, we make use of average session duration and lexical similarities between within-session activities similar to the work in (Faruqui et al., 2015) which used a semantic relational graph obtained from WordNet<sup>2</sup> to enhance existing word embedding. Another important contribution of our proposed method is that we are able to empirically demonstrate that our method is more effective in extracting cross-session tasks without the application of any external information for estimating task relatedness, as was done in (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017a).

Since there is no publicly available log of desktop activities of different users where user tasks have been annotated across different sessions, we use queries from an existing search query log for our investigation of cross session task extraction, using search is an example of a user activity. Essentially we extract search tasks from multiple search sessions. As discussed in Section 3.1 in Chapter 3, each type of activity in our research has a bag of words representation. Hence our proposed approach is sufficiently general that it can be applied to group any kind of user desktop activity log into user tasks.

The rest of the chapter is organized as follows. In Section 5.2 we overview previous work on task extraction and query embedding. In Section 5.3, we introduce our semantic context driven transformation-based word vector embedding algorithm to enhance cross-session query similarity matching. Section 5.4 then describes how the transformed query vectors are clustered into search tasks. Section 5.5 describes an experimental setup to investigate cross-session similarity matching using our approach. Section 5.6 presents the results of our experiments and Section 5.7 concludes explaining how we apply our findings in the context of proactive information

---

<sup>2</sup><https://wordnet.princeton.edu/>

retrieval.

## 5.2 Task Extraction and Query Embedding

In this section, we review existing work in task extraction and embedding approaches and contrast this with our proposed method. There is no work on general task extraction from user activities. Most of the existing work focuses on extraction of search tasks from query logs (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017b; White and Morris, 2007; Hassan Awadallah et al., 2014). For this reason, our review of existing work concentrates only on extraction of search tasks. We first consider existing work on unsupervised task extraction and then work on supervised task extraction.

### 5.2.1 Unsupervised Methods

A method for extracting tasks from search sessions is proposed in (Lucchese et al., 2013). A key question explored in this work is the time gap between related queries associated with a single task or search session. To determine this optimum time gap between query pairs beyond which a query pair does not belong to a single session, they analyzed the time gap between consecutive query pairs from an existing search query log. It was found that 84% of the consecutive query pairs occur within 26 minutes. So they determined if the time gap between a query pair is less than 26 minutes, they are taken to belong to the same session. Following this work, we also use a threshold of 26 minutes in our work on search task extraction. This study also investigated a number of clustering techniques to group together related queries from each session into tasks. A wide range of features were investigated to define the similarity between a pair of queries, e.g. edit distance, cosine-similarity and Jaccard coefficient of character level trigrams. As described in the introduction to this Chapter, that word level similarity features may not capture the semantic similarity between a pair of queries. Hence in contrast to (Lucchese et al., 2013; Wang

et al., 2013), we investigate the use of embedded query vectors to compute similarity, rather than depending on character and word level lexical similarity features, e.g. edit distance, term overlap, trigram character overlap etc. Another difference between our method and the work presented in (Lucchese et al., 2013) is that instead of restricting clustering to each session, we cluster the entire dataset globally. Associated with removing this constraint, we evaluate the effectiveness of clustering the entire dataset rather than on aggregating clustering effectiveness separately for each session as in (Lucchese et al., 2013).

A search task can have one or more sub-tasks. Extraction of hierarchies of sub-tasks within a task was investigated in (Mehrotra et al., 2016). They used a dataset in which queries related to different tasks had been annotated. Given a set of a task related queries, they first estimated the importance of a query term corresponding to the task. A Chinese Restaurant Process (CRP) based posterior inference process was then used to extract sub-tasks from the set of task related queries. In an extension of this work (Mehrotra and Yilmaz, 2017a), the authors proposed a Bayesian non-parametric approach for extracting sub-tasks from a given task.

The main difference between our approach and that reported in (Mehrotra et al., 2016; Mehrotra and Yilmaz, 2017a) is that our focus is on finding cross-session tasks from a query log, rather than finding sub-tasks from a given task. Further, we also show that instead of using similarities between embedded query vectors as one of the features to estimate the relatedness between two queries, like Mehrotra and Yilmaz (2017a), we show our proposed task semantics driven embedding technique for transforming a query in close proximity to its task-related counterpart, to be more effective.

The work in (Verma and Yilmaz, 2014) used the hypothesis that if the same entity is present in a pair of queries, then it is likely that the queries will be related to the same task. Thus an entity extraction method was applied on all the queries to estimate similarities between queries for the purpose of task extraction. A major



disadvantage of this approach is that it relies on the fact that there will be entities present in queries, when in practice there can be queries where no entities are present. In contrast to this, our method does not rely on the entities and an entity extractor to match cross-session tasks.

### 5.2.2 Supervised Methods

A supervised approach for automatically segmenting queries into tasks is proposed in (Jones and Klinkner, 2008). In this work train logistic regression models are trained to determine whether two queries belong to the same task. The study in (Wang et al., 2013) demonstrated that the disadvantage of using a classifier based approach (e.g. (Jones and Klinkner, 2008)) for extracting tasks is that with the binary predictions of the classifier, it is difficult to model transitive task dependence between the queries, e.g. if query pairs  $(q_1, q_2)$  and  $(q_2, q_3)$  are predicted to be part of the same task, the classifier may not predict that  $q_1$  and  $q_3$  are also a part of the same task. The limitations of (Jones and Klinkner, 2008) are alleviated in the work reported in (Wang et al., 2013), which employs a structural SVM framework for estimating the weights of different lexical features to measure the similarity between two queries. They showed that a structural SVM is able to capture the transitive relation between a pair of queries which a logistic regression based classifier can not capture.

The difference between the studies reported in (Jones and Klinkner, 2008; Wang et al., 2013) and our work is that we propose a completely unsupervised approach for clustering queries. This means that our method does not rely on the availability of training data, the construction of which requires considerable manual effort.

Based on the previous work discussed above, we now present our proposed approach for extracting tasks from activities.

## 5.3 Embedding Terms from User Activities

Recall from Chapter 2 that we introduced the concept of word embedding to obtain a semantic representation of a word. The word2vec approach (Mikolov et al., 2013a) aims to create similar vector representations of words that have similar context, and are thus assumed to be significantly semantically related. In this section, we explain why the standard word2vec method may not be suitable for embedding user activities in an abstract space of task semantics for the purpose of using these vectors to extract cross-session tasks. To address this problem, we propose a method of word embedding that is able to capture larger semantic contexts for better estimation of the word vectors.

### 5.3.1 Problems with Short Documents

As discussed in Section 3.1 in Chapter 2 within our research, each user activity is represented by a bag of words and each activity has a maximum word limit (e.g. 10 or 15). The maximum word limit is generally quite low compared to that of a document where the word2vec algorithm can be trained in such a way that it can capture the word semantics present in the document. Hence in our research, activities can be considered as being equivalent to short documents. The word2vec algorithm respects document boundaries by not extending the context vector across them. In the context of our empirical study, we aim to learn word vector embedding from an activity log, where each document in the ‘word2vec’ terminology refers to a single activity. The problem in learning embedding from short documents is lack of context. We first describe briefly the working principle of word embedding to demonstrate its problem with short documents in more detail. Let  $\mathbf{w} \in \mathbb{R}^d$  denote the vector representation of a word  $w \in V$ ,  $V$  and  $d$  being the vocabulary and the dimension of the embedded vectors, respectively. Let  $W$  be a  $d \times V$  matrix, where each  $d$  dimensional column vector represents a word vector. Let  $D$  be an indicator random variable denoting semantic relatedness of a word with its context. Given a

pair of words,  $(w, c)$ , the probability that the word  $c$  is observed in the context of word  $w$  is given by  $\sigma(\exp(-(\mathbf{w} \cdot \mathbf{c})))$ . Word embedding for a given corpus is obtained by sliding a window along with its context through each word position in the corpus maximizing the objective function, shown in Equation 5.1.

$$J(\theta) = \sum_{w_t, c_t \in D^+} \sum_{c \in c_t} \log(P(D = 1 | \mathbf{w}_t, \mathbf{c}_t)) - \sum_{w_t, c'_t \in D^-} \sum_{c \in c'_t} \log(P(D = 1 | \mathbf{w}_t, \mathbf{c}'_t)) \quad (5.1)$$

In Equation 5.1,  $w_t$  is the word in the  $t^{th}$  position in a training document corpus,  $c_t$  is the set of observed context words of word  $w_t$  within a word window,  $c'_t$  is the set of randomly sampled words from outside the context of  $w_t$ .  $D^+$  denotes the set of all observed word-context pairs  $(w_t, c_t)$ , while  $D^-$  consists of pairs  $(w_t, c'_t)$ .

In the case of short documents, which we take to mean that they are comprised of only a few words (e.g. 4 or 5), the average number of context vectors is much lower than the number of contexts available for the standard word embedding scenario of full length documents, e.g. news articles or web pages. Consequently, this may result in ineffective estimation of the word-context semantic relations for the activities. As described in Section 3.1 of Chapter 3, the user activities used in our research scope are also expressed using a small number of words (i.e. essentially they have the characteristics of short documents). Hence embedding user activities faces problems similar to those associated with embedding short documents.

### 5.3.2 Word Vector Transformation with Semantic Contexts

To alleviate the problems of short text contexts when embedding queries, we propose to learn a transformation matrix to transform a set of word vectors to, generally speaking, another abstract space. The aim is to transform a word vector  $\mathbf{w}$  so that it is close to a set of other words that *respect* the characteristics of this abstract space. In the context of our problem, the abstract space refers to an embedding space of task-relatedness with characteristics that queries which are a part of the same search task should be embedded close to each other.

We adopt a general terminology of referring to the desired similarity as *semantic similarity*, which in the context of our problem refers to task-relatedness and should not to be confused with linguistic semantics. Formally, the set of words similar to the word  $w$  is represented by the set  $\Phi(w)$  shown in Equation 5.2,

$$\Phi(w) = \{v : (w, v) \in S\}, \quad (5.2)$$

where  $S$  denotes the semantic relation between a pair of words. In particular, the set  $\Phi(w)$  depends on the definition of the semantic relation  $S$  between two words, which we will describe in Section 5.3.3.

Assuming the existence of a pre-defined semantic relation  $S$  between word pairs, we define the loss function for a word vector  $\mathbf{w}$  as shown in Equation 5.3.

$$l(\mathbf{w}; \theta) = \sum_{\mathbf{v}: v \in \Phi(w)} \sum_{\mathbf{u}: u \notin \Phi(w)} \max(0, m - ((\theta \mathbf{w})^T \mathbf{v} - (\theta \mathbf{w})^T \mathbf{u})) \quad (5.3)$$

Equation 5.3 defines a hinge loss function with margin  $m$  (set to 1 in our experiments as was done for the hinge loss function in (Frome et al., 2013b)). The loss function is parameterized by the transformation matrix  $\theta \in \mathbb{R}^{d \times d}$ , and is learned by iterating with stochastic gradient descent. The word vectors used in learning the parameter matrix  $\theta$  are obtained using the word2vec skip-gram algorithm. After training, each word vector  $\mathbf{w}$  is transformed to  $\mathbf{w}'$  as shown in Equation 5.4.

$$\mathbf{w}' = \theta \cdot \mathbf{w} \quad (5.4)$$

Informally speaking, the objective function aims to maximize the similarity between two word vectors  $\mathbf{w}$  and  $\mathbf{v}$  that are members of the same semantic context. On the other hand, it minimizes the similarity between the word vector  $\mathbf{w}$  and a word vector  $\mathbf{u}$  randomly sampled from outside its context, as defined by the semantic relation  $S$  of Equation 5.2. In principle, the objective function of Equation 5.3 is similar to the word2vec objective function of Equation 5.1, the difference being

in the definition of the context vector. While the word2vec algorithm relies on an adjacent sequence of words to define a context, in our proposed approach, we rely on a pre-defined set of binary relations between words.

Another analogy of Equation 5.3 can be drawn with the multi-modal embedding loss function proposed in (Frome et al., 2013a), where the words from the caption of an image constitute the notion of the ‘semantic context’ of the image vector used to transform it. For our problem, we make use of this context to associate the task-specific relationship between query words. In the following section, we describe exactly how we construct task specific context (i.e.  $S$ ) in our research scope.

### 5.3.3 Constructing the Set of Task-specific Context

Given a standard word vector representation algorithm, e.g. skip-gram (Mikolov et al., 2013a), we slide a window along a text corpus to get the corresponding word embedding. Hence a word is similar to words that appear around its context window. However, such a simple word window-based context may not correctly take into account the task-specific associations between words since activities related to the same task may not always be consecutive. A graph is a hierarchical structure where we can define any type of relation between a pair of nodes by connecting them with an edge. Existing work (Chein and Mugnier, 2008; Boldi et al., 2008), has used graph-based structures to capture any kind of hierarchical or non-linear relations between entities. We use the neighbours of a graph node to enhance the context of a word in a short document.

With this motivation, we propose a graph-based representation learning framework that, generally speaking, can capture the task based semantics between a pair of words. Figure 5.1 shows an example of a small graph constructed from a pair of consecutive queries appearing in a search session. In a general skipgram architecture the word ‘assassination’ has only the word ‘jfk’ as its context word. Figure 5.1 shows that with the graph structure the word ‘assassination’ can have ‘jfk’, ‘kennedy’ and ‘john’ as its context words. We now describe in detail our graph based context

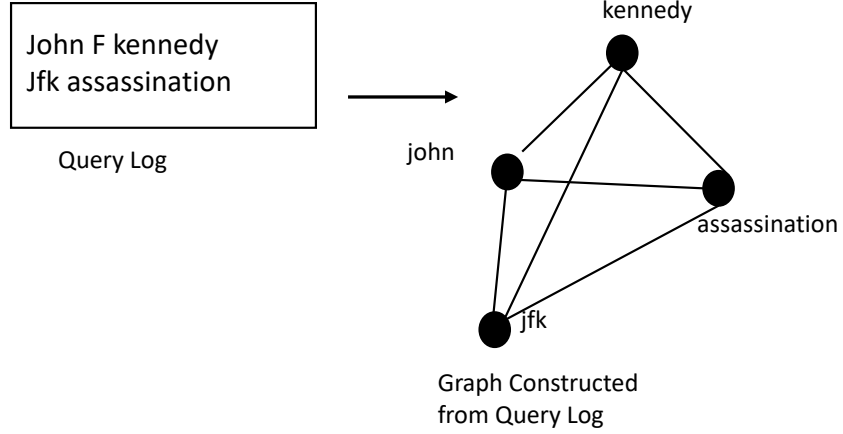


Figure 5.1: Graph Constructed From a Pair of Consecutive Queries.

construction process in detail.

### Graph Construction

We propose to define a graph  $G = (\mathcal{V}, \mathcal{E})$ , where each node corresponds to a word from the vocabulary of the given collection as shown in Equation 5.5.

$$\mathcal{V} = \{x_w : w \in V\}. \quad (5.5)$$

In general, an edge  $(x_u, x_v) \in \mathcal{E}$  represents a *relation* between two words  $u$  and  $v$  of weight  $w(x_u, x_v) \in \mathbb{R}$ . In our research scope, we join a pair of words by an edge if they appear within a single search session ( $Sess$ ) as shown in Equation 5.6.

$$\mathcal{E} = \{(x_w, x_u) : w, u \in Sess\}. \quad (5.6)$$

Since there is a one-one mapping between the set of nodes  $\mathcal{V}$  and the set of words  $V$ , we sample a set of nodes from the neighbours of  $V$  to obtain the context  $S$  for each word. The output of the sampling is the set of nodes which can be used as the context  $S$ . Intuitively speaking, the set of words obtained from the sampling process shows words semantically related to the starting word node. Once  $S$  set is computed for each word, we can get the embedding for each word using Equation 5.3. In the following section we perform the sampling process based on the constructed graph.

### **$S$ set construction**

To construct the set  $S$  for each node in the graph, we first compute the  $\kappa$ -adjacency neighbourhood for each node in the graph as described in Equation 5.7.

$$N_{\kappa}(x_w) = \{x_u \in \mathcal{V} : h(x_w, x_u) \leq \kappa\}, \quad (5.7)$$

In Equation 5.7,  $h(u, v)$  denotes the hop-count or adjacency number between nodes  $u$  and  $v$ . The hop-count between a pair of nodes is the minimum number of nodes which must be traversed to reach from one node to another node. In the general formulation, the set  $N_{\kappa}(x_w)$ , is comprised of the set of nodes reachable from paths of length at most  $k$ , starting at  $x_w$ . This set can act as positive examples (i.e.  $\phi_w$  in Equation 5.3) to learn the embedding of node  $x_w$ . This is because these positive examples seek to make the vector representation of  $x_w$ , similar to the vector representations of nodes in  $N_{\kappa}(x_w)$ .

Instead of defining the set  $S$  as the entire set of the  $\kappa$ -neighbourhood  $N_{\kappa}(x_w)$  of a node  $x_w$ , we take a subset of  $l$  samples drawn from the neighbourhood based on the edge weights. If we consider all the nodes within  $\kappa$ -neighbourhood, this can introduce noise in the embedding process since it will give equal priority to all the neighbouring nodes. We set the likelihood of sampling a node  $x_u$  from the neighbourhood set proportional to the weight of the edge  $(x_w, x_u)$ , i.e.,  $\omega(x_w, x_u)$ . This way of defining  $S$  allows the algorithm to make use of the edge weights in learning the word node representations, i.e. assigning more importance to associations with higher weights in seeking to embed the current word-node close to them.

Our idea, in general, is to use stratified sampling, where each stratum corresponds to a neighbourhood of particular length. The priors assigned to the strata in increasing sequence of adjacency length form a decreasing sequence, which means that the most emphasis is put on direct co-occurrence evidence (i.e. the 1-adjacent neighbourhood), rather than on the 2-adjacent nodes and so on. Stratified sampling requires the strata to be mutually disjoint of each other. This means that

the  $\kappa$ -neighbourhood of Equation 5.7 needs to be redefined to ensure that any node belongs to exactly one of the partitions (defined by its hop-count). To state this formally, we define the set of nodes of (*not up to*) hop-count  $j$ , as shown in Equation 5.8.

$$H_j(x_w) = \cup\{x_u : h(x_w, x_u) = j\} \quad (5.8)$$

The  $\kappa$ -neighbourhood is then defined as described in Equation 5.9. Intuitively speaking if a node appears in both the  $i^{th}$  hop count and the  $j^{th}$  hop count, then we keep only the occurrence from  $i^{th}$  count where  $i \leq j$ .

$$N_\kappa(x_w) = \cup_{j=1}^\kappa (H_j(x_w) - \cup_{j'=1}^{j-1} H_{j'}(x_w)). \quad (5.9)$$

A subset of size  $l$ , comprised of stratified samples from  $N_\kappa(x_w)$ , is then sampled with decreasing priors  $\beta_1, \dots, \beta_\kappa$ , i.e.,  $\beta_j < \beta_{j-1} \forall j = 2, \dots, \kappa$  and  $\sum_{j=1}^\kappa \beta_j = 1$ .

Putting things together, the probability of sampling a node from the set  $N_\kappa(x_w)$ , defined as per Equation 5.9 is then given by Equation 5.10.

$$P(x_u | N_\kappa(x_w)) = \beta_j P(x_u | H_j(x_w)) = \beta_j \frac{\omega(x_w, x_u)}{\omega(x_w, .)}, \quad (5.10)$$

In Equation 5.10,  $\omega(x_w, x_u)$  are edge weights and  $\omega(x_w, .)$  denotes the sum of edges emanating from node  $x_w$ . We now turn our attention to the computation of edge weights (i.e.  $\omega(x_w, x_u)$ ).

### 5.3.4 Edge Weight Computation

We broadly categorize edge weight computation in two different approaches.

#### *Temporal Weight Computation*

In the context of query logs, temporal similarity is likely to play an important role in topically grouping queries. This is because queries in the same search session are usually related to the same topic, as observed in previous studies (Lucchese et al.,



2013; Wang et al., 2013). For example, it can be observed from the AOL query log that the words ‘reb’ and ‘vodka’ belong to the same search session as the words ‘eric’ and ‘harris’ (see the example in the Introduction section to the Chapter). In this case, the semantic relationship  $S$ , as described in Section 5.3.2, considers terms  $u$  (e.g. ‘vodka’) and  $v$  (e.g. ‘harris’) from the same query session to be semantically related.

To define the semantic relation  $S$ , we take into account a temporal context specified by a time window of 26 minutes as described in (Lucchese et al., 2013). Specifically, if two queries belong to the same search session, as defined by a fixed length time window, then each constituent word pair within them is considered to be a member of the set  $S$ .

The intention of the edge weights is to capture the co-occurrence likelihood between two objects (words/queries) within the same session. Hence the weight of a word-word or activity-activity edge is set to the average of the co-occurrence likelihoods across all sessions, as shown in Equation 5.11.

$$\omega(u, v) = \frac{1}{|S|} \sum_{s \in S} \frac{f(u, v|s)}{f(u|s)f(v|s)} \quad (5.11)$$

where  $s$  denotes a session,  $f(u, v|s)$  denotes the number of times objects (word/query)  $u$  and  $v$  co-occur within a session  $s$ . The higher the value of  $w(u, v)$  in Equation 5.11, the higher is the probability of co-occurrence of a pair words or activities within a session.

### ***Temporal-lexical Weight Computation***

In real-life settings, even within a session of a specified time length, users often multi-task their activities (possibly by using multiple browser tabs or windows) (Lucchese et al., 2013; Wang et al., 2013; Mehrotra and Yilmaz, 2017a). Hence not all the queries appearing within a search session may be related to same task. To address this issue, we further modify the edge weights based on both temporal and lexical

similarity between a pair of nodes in the graph. The modified weight formula is shown in Equation 5.12. In Equation 5.12,  $c(u, v)$  measures the lexical similarity between a pair of nodes. If the node is a word-word pair, then  $c(u, v)$  computes the probability of co-occurring both  $u$  and  $v$  in an activity. If the node is an activity-activity pair, then  $c(u, v)$  computes the ratio of the number of word overlaps between the activity pairs to the total number of words present in the activity pair. Thus if an activity  $a_i$  has 5 words in it and an activity  $a_j$  has 10 words in it, and the number of word overlaps between  $a_i$  and  $a_j$  is 4, then  $c(u, v)$  will be equal to  $\frac{4}{11}$ .

$$\omega(u, v) = \frac{1}{|S|} \sum_{s \in S} \frac{f(u, v|s)}{f(u|s)f(v|s)} * c(u, v) \quad (5.12)$$

## 5.4 Clustering of Embedded Query Vectors

In this section, we describe our unsupervised approach to identifying cross-session tasks by clustering query vectors, where the constituent query word vectors are obtained using the word embedding approaches described in Section 5.3.

### 5.4.1 Query Vector Embedding

We hypothesize that the modified word vector embedding approach of Section 5.3.2 will be more effective than existing session extraction approaches (i.e. (Lucchese et al., 2013)) in capturing the session specific semantics of query terms since it takes into account the temporal context of query session information from query logs. Once we obtain word vectors using a standard embedding approach or our proposed transformed embedding approach, we adopt a standard word vector combination method to embed query vectors.

Because of the compositionality property of word vectors (Mikolov et al., 2013a), the simple method of averaging over the constituent word vectors has been reported to work well for various tasks such as term re-weighting and query reformulation (Zheng and Callan, 2015; Grbovic et al., 2015), and we thus adopt this approach

here.

### 5.4.2 Clustering of Query Vectors

Unlike previous approaches of grouping together queries according to the time gap between query pairs, and then clustering the queries within each group separately (Lucchese et al., 2013; Wang et al., 2013), we take a more general approach of clustering the overall set of query vectors both within and across sessions.

Since the number of query clusters cannot be known a priori, the number of clusters is estimated by adopting a clustering approach that does not require the number of clusters to be specified. We adopt the best performing clustering method identified in (Lucchese et al., 2013), referred to as  $QC_{WCC}$ . The study in (Lucchese et al., 2013) used a bag of words representations for queries to cluster them into tasks. For our word embedding based approaches we use query vectors obtained from word embedding approaches to cluster them into tasks. The clustering method uses a graph-based algorithm that extracts the weighted connected components of a graph after constructing a complete graph and then pruning off the edges that are below a predefined threshold,  $\eta$ .

In  $QC_{WCC}$ , the weights between the graph edges are defined by a linear combination of two types of similarities: i) content-based ( $Sim_c$ ), and ii) retrieval based ( $Sim_r$ ), as shown in Equation 5.13, in which the overall similarity is controlled by a linear combination parameter  $\alpha$ .

$$Sim(q_i, q_j) = \alpha Sim_c(q_i, q_j) + (1 - \alpha) Sim_r(q_i, q_j) \quad (5.13)$$

- *Content-based* similarity ( $Sim_c$ ): Measured with the help of character trigrams and normalized Levenshtein similarity between query pairs.
- *Retrieval-based* similarity ( $Sim_r$ ): Each query is contextualized with a Wikipedia collection. More specifically, the top 20 documents from Wikipedia are retrieved. To compute the retrieval based similarity, the Jaccard score is com-

puted between the top 20 ranked lists for a pair of queries. The higher the Jaccard score, the more likely it is that the query pair will be similar.

The edge weights of our graph-based clustering refer to the cosine-similarity values computed between the embedded query vectors and the cosine similarity values between the vectors obtained from the top 10 documents retrieved from Clueweb12B, a publicly available web collection,<sup>3</sup> rather than Wikipedia as was used in (Lucchese et al., 2013).

Our reasons for using Clueweb12B are as follows. Firstly, our study is carried out using queries from a Web search log and hence it is reasonable to expect that a web collection will provide better estimates of semantic similarities between the queries than Wikipedia. Secondly, we observed that the search queries in our dataset are focused (i.e. not on a general or broad topic).

In general Wikipedia is useful if the topics are broad in nature (i.e. have multiple sub-topics in them). Hence the number of matching Wikipedia articles would be expected to be low for our queries due to vocabulary mismatch. On the other hand, the web collection, being very large and diverse, is expected to retrieve more matching articles for these types of queries.

The objective of clustering is to group queries belonging to the same search task together. Clustering is typically evaluated with the effectiveness of the pair-wise decisions of assigning data points to the same or different cluster labels. In our case, the number of true positives is given by the number of query pairs in the groundtruth (i.e. The task annotated dataset), that are judged to belong to the same task, and are also predicted by the clustering algorithm to be a part of the same task. Similarly, we compute the false positives as the number of query pairs which were predicted by the algorithm as a part of the same task, but according to the groundtruth they do not belong to the same task. The true negatives are computed as the number of query pairs which are not part of the same task according to groundtruth and they are also predicted by the algorithm as not being

---

<sup>3</sup><http://boston.lti.cs.cmu.edu/clueweb12/>

User Id	Query	Time Stamp
636926	Robert Francis Kennedy	2006-03-18 08:02:58
3458083	President john f kennedy assasination crime scene	2006-03-17 21:19:29
1604008	John f kennedy	2006-03-18 08:02:58
3271790	Kennedy brothers	2006-03-18 08:03:09

Figure 5.2: Sample queries from search sessions from the AOL query log.

part of the same task.

## 5.5 Experimental Setup

In this section, we describe our experimental study of cross-session extraction of search tasks. We begin with an overview of our datasets, then introduce the experimental baselines used and the objectives of our experiments, finally we set out the parameter settings used in our experiments.

### 5.5.1 Dataset

Similar to previously reported studies (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017a; Mehrotra et al., 2016; Verma and Yilmaz, 2014), we use the AOL query log for our experiments. AOL is a publicly available anonymized search query log. It has information about the user typed queries, user ids and corresponding query time stamps. Figure 5.2 shows an example snippet from the AOL query log. The AOL search log data was collected from 1st March 2006 for three months and contains around 21 million search queries. While the dataset can be considered to be old, recent studies including (Wang et al., 2013) show that web search models show similar trends in both recent commercial search engine query logs and AOL. Hence it is reasonable to say that users’ web search patterns have not changed much since

Query	Task Label
willaim kennedy smith	1
michael lemoyne kennedy	1
robert francis kennedy	1
martha moxley	2
murder in Greenwich	2

Figure 5.3: Sample task labels from the AOL query log.

the collection of this dataset, and thus it is reasonable to use it in our investigations which form part of this research.

For our graph-based embedding framework, we first construct a graph from the whole AOL query log. Since there are 21 million queries in the AOL log, the constructed graph has a large number of nodes. Hence we employ a frequency threshold to only create nodes for words with collection frequency higher than this threshold across the 21 million queries to reduce the size of graph, to make the computation faster. We examined with threshold values from 1 to 5 in the interval of 1, and found that the graph constructed from the threshold value 5 had reasonable size, where graph construction and embedding can be completed in a feasible time. The resultant graph had 1 million nodes and 92,447 edges. If the graph size had been larger, then the only difference would be the presence of some rarely occurring words in the query log in the embedding space.

To construct  $S$  set (described in Section 5.3.3) from the AOL graph, we sampled  $l$  nodes from the 2 hop neighbours of a particular node. We also need  $ns$  the number of negative samples per  $S$  set for the word2vec transformation function described in Equation 5.3. Figure 5.4 shows the effect of varying the sample size on the effectiveness of word embedding in task extraction. From figure 5.4, it can be observed that the optimum FScore (described in Section 5.5.3) is observed when  $l = 5$  and negative sample (i.e.  $ns$ ) size is 10.

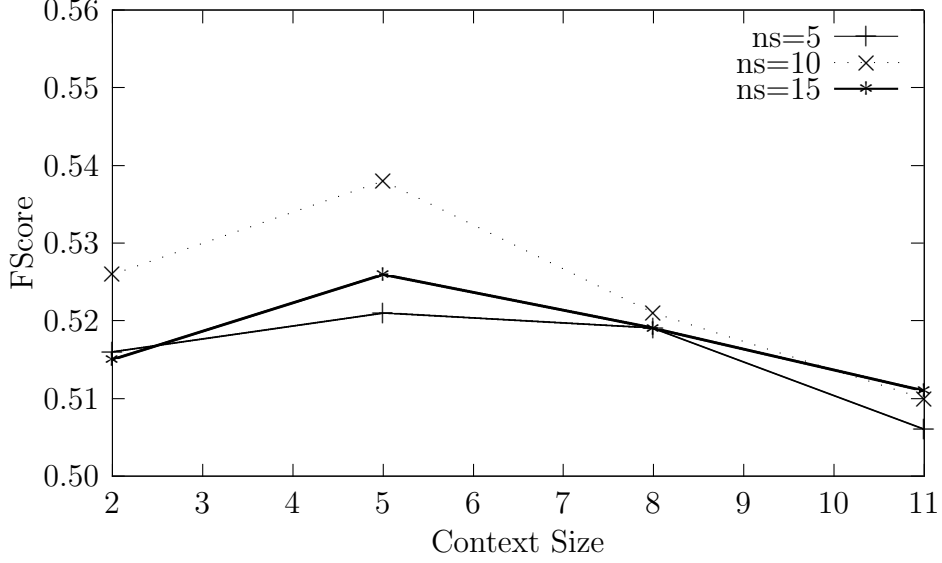


Figure 5.4: Sensitivity of context size with negative sampling for our proposed temporal-lexical word embedding.

To test our embedding approach, we use a subset of 1,424 queries from the AOL query log for the evaluation of task extraction effectiveness, as used in (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017b). The study in (Lucchese et al., 2013) manually annotated these 1,424 queries from 307 sessions from the AOL query log with task labels. An examples of a few task labels are shown in Figure 5.3. Since the purpose of this earlier study in (Lucchese et al., 2013) was only to extract tasks from a single session, in the annotation scheme used in these studies, two queries only qualify as part of the same task if they appear within the same session.

In contrast, since we investigate cross-session task extraction, we re-annotated the task labelled dataset of (Lucchese et al., 2013). In particular, our annotation scheme is solely based on the underlying search intent of the query, not on the session in which it appears. While re-annotating this dataset, our annotators were instructed not to change the task labels within each session, since within a session similar search intent queries already had the same labels as those in the dataset constructed in (Lucchese et al., 2013). Instead, the annotators were asked to re-label task identifiers spanning across different query sessions. For example, in the original annotation ‘robert f kennedy jr’ and ‘robert francis kennedy’ are considered

to belong to two different tasks since these queries were executed in during different sessions. However, our annotation scheme considers them to be a part of the same task since they have the same search intention.

Two persons were employed to carry out our re-annotation of these queries. Initially, both of them separately labelled the same 1424 queries with task labels. Then they were asked to come to a consensus when there was a disagreement regarding the task label corresponding to a query. The annotators were instructed to use a commercial search engine (e.g. Google), if required, to determine if two queries from different search sessions could potentially relate to the same underlying task. Table 5.1 provides an overview of our annotated task labels; this shows that there are a considerable number of search tasks that spanned across session boundaries. It can be seen from Table 5.1 that after post-processing the single session task labels, the total number of distinct tasks is reduced. This is indicative of the fact that the modified dataset is able to consider queries from different search sessions as parts of the same search task (there are 36,768 query pairs which belong to the same task and span across different sessions as shown in Table 5.1). There are 12,124 query pairs in total which belong to the same task and occur within the same session. The post-processed dataset with cross-session task labels that we use for our experiments is publicly available<sup>4</sup>.

### 5.5.2 Baselines and Experiment Objectives

Since our proposed task extraction method is unsupervised, for a fair comparison we only employ unsupervised approaches as baselines. More specifically, we did not consider the approaches reported in (Jones and Klinkner, 2008; Wang et al., 2013) as our baselines since these are supervised approaches. In Section 5.3.4, we described two different edge weight computation methods for our graph based word embedding approach. They are temporal weight computation and temporal-lexical weight computation. Based on these two approaches we get two different embeddings. We

---

<sup>4</sup><https://github.com/procheta/AOLTaskExtraction/blob/master/Task.csv>



Item	Task label granularity	
	Within-Session	Cross-Session
#Queries	1,424	1,424
#Tasks annotated	554	224
#Sessions	307	307
#Sessions with cross-session tasks	0	239
#Query pairs across sessions judged in the same task	0	36,768
#Query pairs within sessions judged in the same task	12,124	12,124

Table 5.1: Dataset statistics of task annotated queries from the AOL query log. Within-Session refers to annotation done by (Lucchese et al., 2013) and Cross-Session refers to our re-annotations as described in Section 6.5.2.

called them as *temporal embedding* and *tempo-lexical embedding*. To demonstrate the potential benefits of our proposed tempo and tempo-lexical word embedding based approaches for the query terms, we employed the following baselines.

1. **QC<sub>WCC</sub>**: As our first baseline, we re-implemented QC<sub>WCC</sub>, the best performing approach in (Lucchese et al., 2013) (briefly described in Section 5.4.2). The study in (Lucchese et al., 2013) investigated a wide range of features, clustering methods and parameter settings. We adopt the same linear combination of similarities in our study as shown in Equation 5.13. (Lucchese et al., 2013) only to extracted search tasks within a search session.

Since our focus is on cross-session task extraction, our re-implementation of this work involves a slight change to the original method, instead of using a Wikipedia document collection, we use ClueWeb12B. (Lucchese et al., 2013) used the top 10 documents retrieved from Wikipedia for each query to compute similarity between two queries. They used a commercial search engine to return the Wikipedia articles corresponding to each query.

To make a reproducible experiment, we use the LM-JM (Language Model with Jelineck-Mercer smoothing) with the smoothing parameter set to 0.6, as suggested in (Lavrenko and Croft, 2001), to retrieve the top 10 documents

from Clueweb12B for each query. Initially, we carried out experiments with the top 5 to 20 documents in the interval of 5. For each configuration we grouped task related queries and observed the Fscore value. We found that the optimum Fscore value was using the 10 documents, so we report results using top 10 documents in Table 5.3.

2. **Qry vec skip-gram:** In this approach, query vectors are obtained by summing over the constituent word vectors obtained using the standard skip-gram model on the AOL query log dataset (Mikolov et al., 2013a). We did our experiments with 100 and 200, 300 and 400 dimensional vectors, but observed the optimum Fscore value with 300 dimensions. Hence we report the results in Table 5.3 only with 300 dimensions.

For training the skip-gram model on the AOL corpus, we used a window size of 5 and negative sample size of 10. For window size and negative sample size we varied the parameters, and found 5 and 10 to be optimal. The cosine similarity between a pair of query embedding is used to group the queries into tasks.

3. **Qry vec (Pre-trained Google news vectors)** In this approach, rather than using skip-gram word vectors trained on the AOL query log, we use pre-trained skip-gram vectors trained on the Google News corpus <sup>5</sup>. As described in Section 5.3.1, word vectors trained on query logs have the problem of short contexts since queries in general comprise of two or three words. Thus we examine a baseline using pretrained embedding to observe how a method performs which uses embeddings trained on a large general corpus where there is no issue of insufficient context vectors.

## Proposed Approaches

Here we discuss about our proposed approaches that we used to compare with our baselines.

---

<sup>5</sup><https://github.com/mmihaltz/word2vec-GoogleNews-vectors>

**Qry vec (All-in-one Session)** Recall from Section 5.3.2 that we hypothesize that additional context is likely to be useful to learn the vector representations of constituent words of short documents (in this case queries). We introduce a method that we name **Qry vec (All-in-one Session)**, to investigate the use of additional context of any length in improving existing embeddings. To provide additional context, we ignore query or session boundaries. So for each word in the query log there is no issue of insufficient context (i.e. The context for a word can cross the boundary of a query or a session).

In addition to all the baselines outlined above, we report the results of our proposed graph-based transformed embedding approaches in Table 5.3. Recall from Section 5.3.2 that we use a sampling process on the 2-hop neighbours of each node of the weighted graph to construct additional context for each word, where the additional context is eventually used to transform the word embedding. We report two methods in Table 5.3 based on the two different edge weight computation process described in Section 5.3.2. The methods are **Qry vec temporal context** (i.e. based on the Equation 5.11) and **Qry vec tempo-lexical context** (i.e. based on Equation 7.2).

The objective of the experiments is to examine how our proposed query term embedding approach performs against the baselines in finding search tasks across sessions, to determine whether within-session adjacency information (i.e. as described in edge-weights in Section 5.3.4) can be useful to learn task specific semantics.

### 5.5.3 Parameters and Evaluation Metrics

Here we examine parameters used in our experimental setup and the evaluation metrics used to measure the effectiveness of our proposed cross-session task extraction approach.

## Parameters

For all the task extraction approaches (i.e.  $QC_{WCC}$  and all the embedding based approaches), once we have a query representation (e.g. bag of words or embedding vector representation) we cluster them using the graph-based clustering approach described in Section 5.4.2.

1.  $\alpha$ : Recall from Equation 5.13 in Section 5.4.2 that there is a parameter  $\alpha$ , that controls the contribution of content-based similarity and retrieval-based similarity for computing the similarity between a pair of queries. In all our methods (i.e. baselines and proposed approaches), we tune  $\alpha$  from 0 to 1 in steps of 0.1.
2.  $\eta$ : The second parameter common to all the methods is the threshold  $\eta$ . As described in Section 5.4.2,  $\eta$  is used in clustering to prune off edges from the weighted similarity graph between query pairs. For a clustering method, we choose the  $\eta$  value for which the method gets the best F-score value. This is described in the following subsection on Evaluation Metrics.

## Evaluation Metrics

Here we introduce the evaluation metrics used to measure the effectiveness of task extraction in our investigation. Since we use clustering to extract search tasks from a sequence of queries, we use standard clustering evaluation metrics to evaluate the effectiveness of the task extraction. Based on the count of true positives, false positives, true negatives and false negatives, we compute the standard metrics of precision, recall, and F-score (similar to (Lucchese et al., 2013)).

Additionally, to measure how many of the total number of queries that are part of the same search tasks discovered by a particular approach, we compute the cross-session recall (denoted as ‘CS-Recall’). This metric is computed as the ratio of the number of correctly identified cross-session similar-task query pairs against the total number of cross-session similar task query pairs (36,768 as reported in Table 5.1).

The formula for cross-session recall is described in Equation 5.14. Let  $CS-TPQP$  denote the actual number of cross-session query pairs that belong to same search task. Let  $CS-QP$  denotes the number of cross-session query pairs that were correctly predicted to be a part of the same search task by the clustering algorithm.

$$CS-Recall = \frac{CS-QP}{CS-TPQP} \quad (5.14)$$

In order to extend evaluation of our proposed approach to within-session task extraction, for comparison with existing studies, we compute the clustering metrics for each individual session, and then compute the weighted average of these values over each session, as reported in (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017a). Although these earlier studies refer to this weighted measure as F-score, we refer to this version of F-score as ‘Session-F-score’.

## 5.6 Experimental Results

In this section, we report the results of our investigations of our proposed embedding-based cross-session search task extraction in comparison to our identified baselines. We first investigate the effectiveness of our proposed approach on cross-session search task extraction, and then report and compare results with existing approaches for within-session task extraction. First, we show the effect of varying the parameters  $\alpha$  and  $\eta$  separately in Figure 5.5.

We varied  $\alpha$  and  $\eta$  for the methods  $QC_{WCC}$ , ‘Qry-vec skipgram’, ‘Qry-vec temporal’ and ‘Qry-vec tempo-lexical’ in the test set of 1,424 queries. From the three standard embedding based approaches in Table 5.3 (i.e. ‘Qry-vec skipgram’, ‘Qry-vec-all-session-in-one’, ‘Qry-vec-pretrained’), we choose the best one in Figure 5.5 to examine variations in  $\alpha$  and  $\eta$ , so we do not consider ‘Qry-vec-pretrained’ and ‘Qry-vec-all-session-in-one’ in the plot. The optimum values of  $\eta$  for each method in the bottom graph of Figure 5.5 correspond with those reported in Table 5.3. Similarly, for the plot at the top of Figure 5.5, the optimum  $\alpha$  values correspond to those

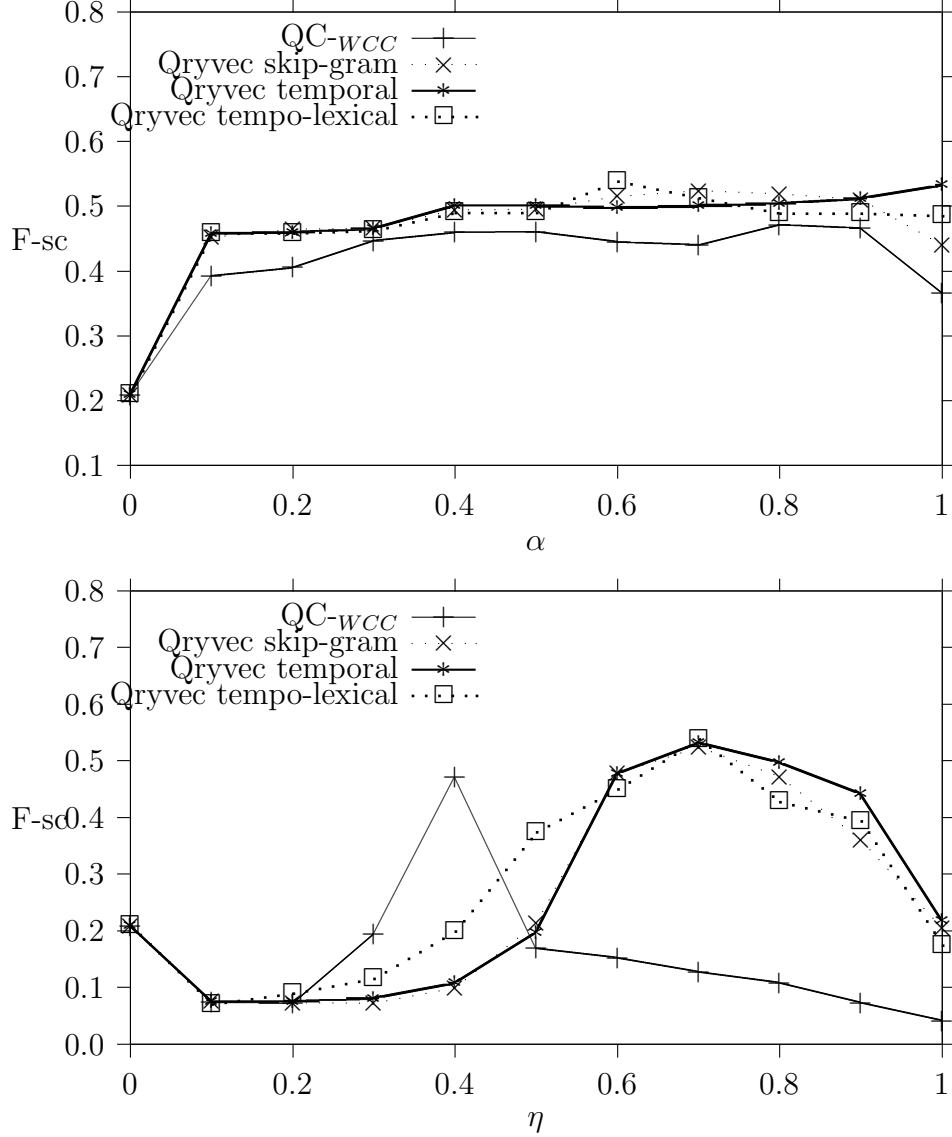


Figure 5.5: Sensitivity of task clustering with variations in  $\alpha$  (top) and  $\eta$  (bottom).

reported in Table 5.3. Since the training set had no task labels it was not possible to tune the parameters  $\alpha$  and  $\eta$  in the training set. It can be observed in Figure 5.5 that for most of the values of  $\alpha$  and  $\eta$  the proposed approach performs better compared to the other baselines.

A value of  $\alpha = 1$  considers only the content based similarity (see Equation 5.13). It can be observed from Figure 5.5 (top) that at  $\alpha = 1$ , the F-score values for all the query embedding based approaches are higher than the baseline method of QC $_{wcc}$ . This indicates that the query embedding based approaches perform well without relying on similarity-based retrieval using an external collection (i.e. Clueweb12B).

Task extraction method	$\alpha$	$\eta$	Precision	Recall	F-score
QC <sub>WCC</sub> using Wikipedia	0.8	0.4	0.825*	0.802 *	0.812 *
QC <sub>WCC</sub> using ClueWeb12B	0.8	0.4	0.831	0.815	0.834
(Mehrotra and Yilmaz, 2017b)	-	-	0.849	0.835	0.845
Qry vec skip-gram	0.7	0.8	0.834	0.832	0.839
Qry-vec (Pretrained wordvec)	0.6	0.5	0.832	0.818	0.837
Qry-vec temporal context	1	0.7	<b>0.851</b> *†	0.831 *†	<b>0.847</b> *†
Qry-vec tempo-lexical context	0.6	0.7	0.845 *†	<b>0.838</b> *†	0.840 *†

Table 5.2: Comparison between the best results obtained after parameter tuning on different unsupervised approaches of within session task extraction. \*† indicates statistical significance (paired t-test with 95% confidence) with respect to QC<sub>WCC</sub> using Wikipedia, ‘Qry-vec temporal context’ and ‘Qry-vec tempo-lexical context’ respectively.

	Method Name	Parameters		Metrics			
		$\alpha$	$\eta$	F-score	Prec	Recall	CS-Recall
Baseline	QC <sub>WCC</sub> using ClueWeb12B	0.8	0.4	0.471	0.387	0.603	0.1930
	Qry vec skip-gram	0.7	0.8	0.524*	<b>0.465</b> *	0.602	0.7161*
	Qry vec (All-in-one Session)	0.7	0.5	0.499	0.430	0.595	0.6400
	Qry vec (Pre-trained wordvec)	0.6	0.5	0.473	0.410	0.558	0.6400
Proposed	Qry vec temporal context	1.0	0.7	0.536*†	0.461*	0.643*†	0.7393*†
Approach	Qry vec tempo-lexical context	0.6	0.7	<b>0.538</b> *†	0.441*	<b>0.691</b> *†‡	<b>0.7395</b> *†‡

Table 5.3: Comparison between the best results obtained after parameter tuning on different unsupervised approaches of cross session task extraction. For all methods,  $1 - \alpha$  represents the weight of the semantic similarity estimated from ClueWeb12B. \*†‡ indicates statistical significance (paired t-test with 95% confidence) with respect to Lucchese et al. (2013), ‘Qry vec skip-gram’ and ‘Qry vec with temporal context’ respectively.

In general, it can be observed that over a wide range of  $\alpha$  and  $\eta$  settings, the F-score values of the embedding based methods outperform the QC<sub>WCC</sub> method.

### 5.6.1 Within-session task extraction

Our first experiments evaluate our embedding methods against state-of-the-art baselines. The session duration span of 26 minutes, similar to (Lucchese et al., 2013), is used to define query clustering to each individual sessions to clustering. Similar to (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017a), we employ the session averaged clustering metrics to measure the effectiveness of the approaches (see Section

5.5.3). We use the within-session ground-truth of (Lucchese et al., 2013) to evaluate task extraction effectiveness.

Table 5.2 reports the results for our within-session task clustering approaches. The following observations can be made with regard to Table 5.2.

- Firstly, we can see that the use of ClueWeb12B for contextualizing queries (i.e. first row of Table 5.2) contributed to an improvement in task extraction effectiveness compared to the original approach of (Lucchese et al., 2013) (i.e. second row of Table 5.2) which uses Wikipedia. So we can say that our re-implementation of (Lucchese et al., 2013) using ClueWeb12B is comparable or better than that of the original.
- Secondly, an important observation is that the use of word embedding along with contextual information from ClueWeb12B (i.e. third to sixth row) outperforms the approach of trigram and Levenshtein based similarity computation of (Lucchese et al., 2013).
- Thirdly, it can be observed that F-Score improves with the application of transformation based word vector embedding of the query terms compared to ‘Qry-vec skipgram’. In Table 5.4, we show the nearest neighbours for the word ‘jfk’ before and after transformation. In the left column of Table 5.4 we can observe that the neighbour queries belong to only the ‘jfk airport’ travel related area. In the right column, of Table 5.4, we can observe that the neighbour queries belong to the topic of assassination of president john f kennedy and jfk airport. So we can say that right column in Table 5.4 (i.e. using our proposed embedding technique) captures the different tasks related to the word ‘jfk’ compared to the left column (i.e. standard skipgram). This can be the potential reason for which our proposed embedding approach performed better compared to ‘Qry-vec-skipgram’.
- Forthly, our proposed word embedding approach (i.e. ‘Qry-vec temporal context’) also outperformed existing task extraction method within a single session



Word	Nearest Neighbors	
	Before Transformation	After Transformation
jfk	jfk airport	jfk airport
	kennedy airport	john f kennedy
	USA	jfk assassination
	travel	president

Table 5.4: Nearest Neighbors for the word ‘jfk’

(Mehrotra and Yilmaz, 2017b).

The temporal context proves to be more effective than the tempo-lexical one in terms of Session-F-score. One reason for this is that in most of the sessions a user was involved in a single task within a session. Temporal context utilizes this fact, whereas tempo-lexical context cannot capture the overall context because it considers only lexically similar queries within a session as context.

### 5.6.2 Cross-Session Task Extraction

Table 5.3 shows the results of graph-based clustering (as described in Section 5.4.2) applied on all our baselines and our proposed approach with optimal  $\alpha$  and  $\eta$  settings (as described in Section 5.5.3). It can be seen that the first baseline approach  $QC_{WCC}$  performs poorly in terms of FScore, Precision and CS-Recall compared to all other approaches.  $QC_{WCC}$  is the only method reported in Table 5.3, which uses only text based similarity (e.g. trigram and Levenshtein similarities) to compute the similarity between a pair of queries.

All other methods except  $QC_{WCC}$  rely on word embedding to compute similarity between a pair of queries. So we can say that embedding based similarity captures the task based semantic similarity between a pair of queries better than use of text-based similarity features.

It can also be observed from Table 5.3 that ‘Qry vec (All-in-one Session Context)’ yields worse results in terms of F-score, Precision, Recall and CS-Recall than the Qry vec skip-gram approach. ‘Qry vec (All-in-one Session Context)’ does not use

any session or query boundary to obtain embedding of query terms, whereas ‘Qry vec skip-gram’ treats each query as a separate document and the context window of a word never slides across the query. Hence we can say that a focused context performs better for embedding the query terms for task extraction compared to using a large context without any query boundary.

Results with word vectors pre-trained on a large news corpora (i.e. the approach ‘Qry vec (Pre-trained word vector)’, 4th row in Table 5.3), show that additional out-of-domain and generic context is not helpful for improving the quality of the embedded query term vectors compared to ‘Qry vec skip-gram’ and ‘Qry vec (All-in-one-Session)’.

Transformation of the word vectors leveraging the semantic contexts (i.e. fifth and sixth rows of the Table 5.3) outperforms the baseline approaches in terms of all the metrics (i.e. F-score, Precision, Recall, CS-Recall). Recall from Equation 5.13 in Section 5.4.2, that we use top 10 Wikipedia articles retrieved corresponding to a query to contextualize a query.

The most important observation in Table 5.3 is that the use of temporal context in learning word vectors results in best performance for  $\alpha = 1$ , i.e. when no retrieval-based similarity is used (see Equation 5.13). This suggests that optimally trained word vectors can produce effective task clusters without the use of external collections in contextualizing the queries. A potential reason for this observation is that rather than an external collection, task related words are more likely to be present in the extra context (i.e.  $S$  in Equation 5.4) that we are providing for each word to transform the word embedding.

The use of tempo-lexical contexts, i.e. when the semantic context used to learn the transformation matrix for the word vectors is restricted to similar queries within search sessions, the clustering effectiveness improves further in terms of all the metrics. In particular, Table 5.3 shows that both tempo and tempo-lexical transformations are able to improve Recall significantly, suggesting that the transformation helps to group more truly task-related queries into the same cluster.

## 5.7 Concluding Remarks

In this chapter, we addressed our second research question relating to the identification of similar user activities related to common information goals. We proposed a transformation based word embedding approach that takes into account the temporal and tempo-lexical contexts of activities to learn task-specific semantics. Our experiments on the AOL query log indicate that our proposed temporal and tempo-lexical embedding method outperforms the baseline word2vec embedding and other approaches. The experimental results confirm that we can use our proposed graph-based embedding method to address RQ2 which is to group user activities that are related to similar information goals or tasks. Further analysis of our proposed graph based embedding technique on standard embedding evaluation tasks are shown in Appendix C.

Once we can identify activities similar to a user’s current activity using our proposed embedding approach, we can use these similar activities to formulate proactive queries using the proactive query formulation method framework described in Chapter 3. We examine the use of embedding approaches for proactive query formulation in Chapter 6 and 7 for single-stage and multi-stage task scenario respectively. Since Chapter 6 focuses on a simulated task scenario where there user activities only relate to the user’s current task, standard embedding are expected to work well, and there is no need to group similar activities from a sequence of activities. In Chapter 7, we specifically use the embedding approach proposed in this chapter to group queries in our investigation of a multi-stage task scenario.

## Chapter 6

# Proactive Suggestion For Single Stage Tasks

In this chapter, we address RQ3 (introduced in Chapter 1), which concerns examining proactive formulation of queries for single stage task scenarios. In our research scope, a single stage task refers to those tasks where we have information about only a user’s previous and recent desktop activities, and the task is related to a single information need. For instance, an example of such, a single stage task could, be a user writing a research paper on a topic, such as Retrieval Models in IR. While writing a particular section of this paper, the objective of a proactive IR model in this setting would be to provide the user with relevant articles or papers that they might find useful while writing their research paper.

There is no publicly available dataset of desktop activities suitable for exploring this research question. To address our research question we need a dataset containing information about what the user was reading or writing or clicking. There is also no available single software applications which can do all these different activities (i.e. reading, writing, or accessing a document or application) mentioned earlier. Existing work related to personal information management (Bernstein et al., 2007; Cai et al., 2005; Hu and Janowicz, 2012) which describes how information sources can be organized and activities presented in a user’s desktop can be used to enhance

their desktop search experience. However in this thread of work the information sources present in the desktop need to be explicitly tagged or linked to enhance the user’s experience. Work in (Bernstein et al., 2007) asked the user to explicitly note down their activities. However, explicitly tagging or linking may interrupt a user’s workflow. We wanted to investigate a scenario where the user’s workflow were not interrupted for collecting data. Hence we introduce a simulation framework for this task scenario which enables us to compare alternative potential approaches to proactive search for which results are reproducible. We first describe the background and motivation for our simulation framework, and then describe our experimental setup and our investigation of proactive IR using this framework.

## 6.1 Background and Motivation

In this chapter, we examine the anticipation of a user’s information needs (i.e. formulating queries on the user’s behalf) from his activities, and providing them with potentially relevant information sources. In this investigation we observe the effect of user knowledge on the query formulation process. We use a simulation setup to estimate the effectiveness of the query formulation process.

Existing literature on this topic can broadly be categorized into two different areas:

1. Simulation frameworks for standard IR
2. Associated document retrieval where queries are generated from large documents.

The setup for associated document retrieval is similar to our simulation framework where we use texts extracted from a user’s activities to formulate queries to identify potentially useful documents. Here we review existing literature on each of the above mentioned areas (i.e. simulation and associated document retrieval) separately.

### 6.1.1 Simulation Setup Used in IR

Somewhat similar to our work, simulated users and search agents were investigated by Maxwell and Azzopardi (2016a,b). However, these studies mainly focus on keyword-based automatic query formulations from simulated clicks on documents, and do not address the research questions that we investigate, i.e. those involving user knowledge (initial cognitive state). Initially they used a maximum likelihood estimate to choose a pair of words from all the words appearing in a Title and a Description field of a TREC Topic. As user actions progressed, they used words appearing in clicked documents to identify terms to include in a query. Moreover unlike (Maxwell and Azzopardi, 2016a,b), our proposed simulated interaction method distinguishes between different types of user activities (e.g. reading and writing).

A relatively simple simulated user read activity model was proposed in (Koskela et al., 2018). They chose the terms with the highest tf-idf weights from a document to simulate the key focus of the content that a user is currently reading. However, in a realistic situation it is more likely that a user will read a sequence of words or sentence at a stretch, hence the words read will be contiguous and semantically related. The terms having highest tf-idf values are selected from a whole document. These terms may not be contiguous. Since a user generally reads a sequence of words at a stretch, extracting the most important terms in terms of tf-idf values is not likely to reflect a focused topic encountered by a user while reading a document. This study also reported a controlled user study of essay writing on a specific set of topics with real users which demonstrated that a proactive retrieval system allowing a user to read suggested documents or to save them for later revisits, can help the user to accomplish a task more efficiently. Unlike (Koskela et al., 2018), we generate simulated reading content by selecting sentences from a document which are more focused on a topic than choosing a small number of non-contagious terms from the whole document.

### 6.1.2 Associative Document Retrieval

A particular example of associative document search (Takaki et al., 2004; Chen et al., 2010) is patent search. Previous studies of automated query formulation in patent search have shown the following to be useful: i) assigning different relative importance to different sections of a patent document (Xue and Croft, 2009), ii) subtopic analysis for segmenting a patent document into smaller blocks (Takaki et al., 2004), and iii) a feedback algorithm for reducing the number of terms in queries (Ganguly et al., 2011a).

Existing research in query formulation for other types of documents (e.g. news and web) has investigated: i) an encoder-decoder based neural approach to generate query terms (Han et al., 2019), ii) use of user profiles to formulate personalized queries (Somlo and Howe, 2003), iii) extracting entities and noun phrases for query generation from user selected text (Lee and Croft, 2012), iv) use of different fields of a structured document to formulate queries (Crouch et al., 1990).

Previous research (Lee and Croft, 2012; Crouch et al., 1990) on associated document retrieval suggests that queries formulated from given long pieces of text help to retrieve potentially relevant documents related to the topic of the text. The objective in our single stage simulation setup is to retrieve potentially relevant documents corresponding to a task topic given a user’s activity context. As a result of this in our simulation setup, instead of content generated live by the user, we make use of segments of text extracted from the documents in a static collection as simulated content. Similar to associated document retrieval, the simulated content can also be used to proactively formulate queries on behalf of a user to support them in their current task. A major difference between our work and existing research on associative document search is that existing work has no notion of temporal context in the query generation process, which is an important piece of information in our case. Moreover, the query formulation process in our case involves extracting information from multiple documents rather than focusing on a single piece of text.

## 6.2 Workflow of Proactive Agent for Single Stage Tasks

For a single stage task, we simulate a proactive agent which runs continuously as a service in the background of a computing environment (e.g., a desktop or a mobile device), automatically formulating queries based on a user's activities and then retrieving lists of potentially relevant documents without the user being involved in this process. The user may then be notified through system notifications of any newly discovered potentially relevant documents which are different from what the user has already seen. As discussed in Section 3.1 of Chapter 3, we provide notifications after a fixed number of activity intervals. Continuous proactive suggestion notifications may be distracting. However, in this study, we do not focus on when we should provide proactive suggestions. Rather we investigate whether it is possible to automatically find relevant information sources that may be useful to a user in completing their task.

This workflow setup thus allows provision for the user to click on these suggested documents, and to access knowledge contained in them, which may assist them in accomplishing their task. The click information from these suggested documents can then be further used as feedback information to reformulate queries to seek further relevant information, akin to the concept of click-based relevance feedback in standard active IR. We now describe the general workflow of the interaction between our proactive search agent and a user of a computational system in detail.

### 6.2.1 Interaction Environment

Prior to describing the user actions within a proactive environment, we define the notation for the environment itself and its primary constituents. A hypothetical proactive environment,  $E$ , refers to an interactive computing environment (e.g. a desktop operating system) comprised of input and output processes, such as the file system for reading and writing locally resident files, a browser for reading and



writing remotely residing content (e.g. in the cloud), editor programs for creating new content, etc.

The key constituents of this environment that are useful to a proactive agent comprise a set of *active* entities (e.g. recently accessed files within  $E$ ), which we call the *Active Set* of the environment, as described below.

1. Reading state ( $R$ ): the set of local files within the file system of  $E$  that have been recently accessed (e.g. files that are currently open or have been opened recently), or the set of remote resources (e.g. on the web or in the user’s private cloud) that have been accessed recently( e.g. web pages that are open in their browser).
2. Writing state ( $W$ ): the set of newly-created user content e.g. the content typed by the user in an editor.

As per our research scope, we restrict our attention to a single modality of textual information only, e.g.,  $R$  refers to text extracted from documents (which, generally speaking, could themselves be multi-modal or of a different modality to text). We use the notation  $R_i$  as a user’s *reading state* at time stamp  $i$ .  $R_i$  consists of a set of files/documents that the user has read/accessed upto timestamp  $i$  and the output of an eye tracker while reading these documents.

### 6.2.2 Active Set

In our simulation setup, the Active Set at any instant refers to the set of documents and text snippets that has been accessed by the user up to that time. The Active Set acts as a potential source of information which our proactive agent can exploit to estimate the user’s current information interests associated with their current activities. It is useful to think of the Active Set of an environment as a function of an instant in the user’s interaction timeline. Moreover, the definition of an *Active Set* requires specifying an inclusion criteria, which could either be based on: i) a simple approach, such as time-thresholding (e.g. excluding files that have not been

accessed for too long out of the active set), or ii) a more involved approach, such as employing an eye tracking device (Buscher et al., 2008) to include only those sub documents in the Active Set which the user has actually read in the recent past.

The Active Set at each instant in a user’s timeline of activities (interactions) is well-defined, which we denote as the 2-tuple  $A_t = (R_t, W_t)$ , where  $t$  denotes a time instance, and  $R_t$  and  $W_t$  denote a set of recent documents that the user has recently read and written respectively (as per the inclusion criteria) in the Active Set at time  $t$ . Active Set at the start of the interaction (i.e.  $A_0$ ), consists the set of activities executed by the user in past.

### 6.2.3 User Actions

In our simulation setup, user actions refer to user activities like reading or writing. An Active Set, as described in Section 6.2.2, is constructed from the observed user actions. The content accessed by a user is the result of a user action (i.e. read or write), and is used to support the proactive agent in estimating the user’s recent activity context.

A user  $U$  who starts interacting with a proactive environment,  $E$  at a (relative) time  $t = 0$  progressively makes changes in their Active Set depending on the actions from which they arise. While reading affects the  $R$  component of an active set tuple, writing modifies the  $W$  component. Table 6.1 lists the types of user actions and describes how these actions modify the Active Set within an environment at a time  $t$ .

While some of the user actions in Table 6.1 contribute to adding new member objects to the Active Set (e.g. opening a file or writing text), some remove elements from the set (e.g. closing a file, shifting focus to a different window, etc.).

Table 6.1: Types and examples of user actions that change the active set of an environment.

Action	Active Set change	Example
Open a file $f$	$R_{t+1} \leftarrow R_t \cup f$	Open a PDF file in Acrobat
Read file $f$	$R_{t+1} \leftarrow R_t \cup f$	Click or Alt+Tab to focus on an Acrobat window
Read a part $f'$ of $f$	$R_{t+1} \leftarrow R_t \cup f'$	Content $f'$ is the output of an eye-tracking device
Open a web page $p$	$R_{t+1} \leftarrow R_t \cup p$	Click on a hyperlink from a browser
Write text $d$	$W_{t+1} \leftarrow W_t \cup d$	Composing a mail or typing a query in a search-box
Close a file $f$	$R_{t+1} \leftarrow R_t - f$	Mouse-click or Alt+F4
Attention away from $f'$	$R_{t+1} \leftarrow R_t - f'$	Detected by shift in eye gaze

#### 6.2.4 Proactive Recommendation Agent

With the definition of the interactive environment and its instantaneous user action states (comprised of the Active Sets) in place, we now discuss the working methodology of a proactive recommendation agent that seeks to find relevant information for a user based on his recent activities.

Assuming that a user has conducted a series of recent activities (such as the ones outlined in Table 6.1) to accomplish a task within the environment, a proactive recommendation agent can potentially leverage information obtained from recent Active Sets (which it keeps track of) to estimate the user’s recent topics of interest.

In principle, a proactive agent can make use of a bag-of-words representation obtained from each individual component of a number of most recent Active Sets to formulate a *query* implicitly in the background without the user being aware of the process (Dumais et al., 2004, 2016). This query can then be used to retrieve a list of documents from either the public web or the local file system. For the former, a possible implementation may invoke a search API service with the formulated query passed as an argument for retrieving information, whereas for the latter, one could use one or more local indexes (comprised of textual content extracted from files) to retrieve related files.

Figure 6.1 shows the workflow of a proactive IR model in a real user setup. As

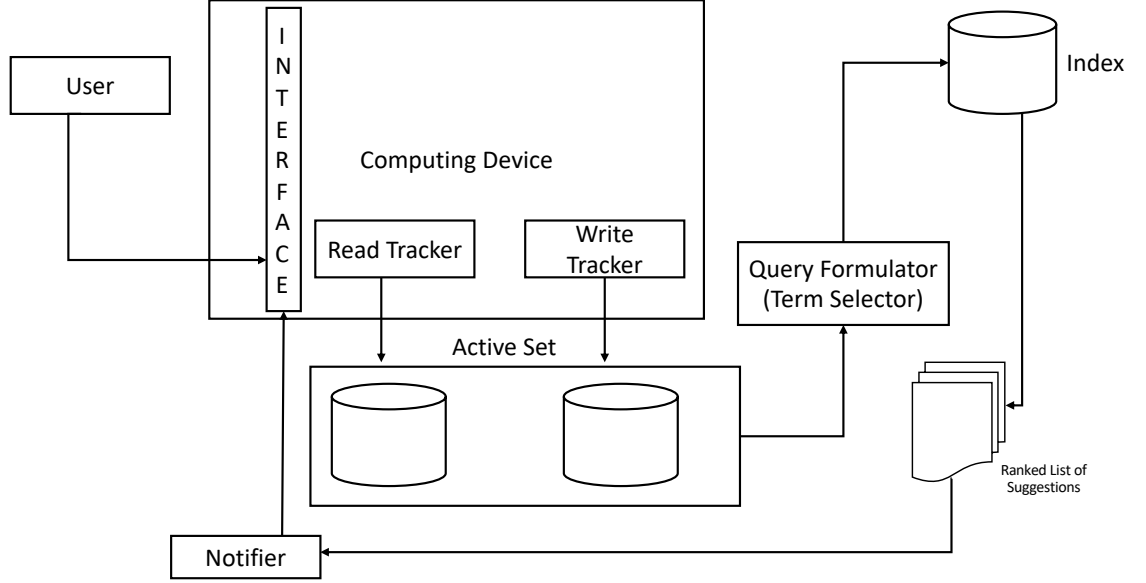


Figure 6.1: Work Flow of Proactive IR model in a Real User Setup.

shown in Figure 6.1, the Active Set is always updated with a user’s recent read or write activities. The proactive agent formulates a query from the updated Active Set (i.e. Query Formulator component in Figure 6.1). The query is then used to retrieve a ranked list of suggestions which is shown to the user through a notifier. Now we will turn our attention to how we can simulate an Active Set for single stage task scenario described in this section.

### 6.3 Simulation Setup With Document Relevance

In this section we describe the components of simulation setup, beginning with a model of user knowledge of the topic of proactive search, then introducing models of user reading and writing activities. In our simulation setup, we use a parameter  $K$  to instantiate a user.  $K$  denotes the knowledge of the user. In terms of the Active Set of a proactive environment definition (see Section 6.2), the initial Active Set depends on the initial knowledge of the user and its subsequent progress over time. Different users will have different levels of initial knowledge. For example, if the initial active set of a user  $U_1(K_1)$  is  $A_0^1 = (R_0^1, W_0^1 = \phi)$  and that of  $U_2(K_2)$  is  $A_0^2 = (R_0^2, W_0^2 = \phi)$ , then  $|R_0^1| > |R_0^2|$  if  $K_1 > K_2$ , or in other words, the first user has acquired a greater

amount of knowledge since he has accessed a greater number of documents (local and remote) than the second. The initial Active Set is important since it will affect the query formulation process. The more content that is present in the initial Active Set, the more information a proactive agent has available to estimate the user's current potential information needs and an opportunity to provide informational support. The user may not have an information need in the sense of needing to look for information, e.g. via an active search, but may benefit from the provision of information relevant to their current activity.

As a laboratory based version of the environment of Section 6.2, a user (parameterized by their knowledge) in the simulated environment conducts two different types of actions, that of *reading* and *writing* towards achieving a goal. In contrast to the wide range of different types of goals in a real environment (e.g. writing a piece of code to solve a computational problem or writing a research paper, etc.), the goal of a user in reading state in this simulated environment is to learn more information about a particular topic of interest. Similarly the goal of a user while writing in the simulation setup is to express their knowledge on a particular topic in writing. To achieve these goals, a simulated user starts with a small set of relevant documents on a topic, *reads* these documents to build an abstract cognitive state inside his head. The user then either *reads* more documents or *writes* pieces of text in an editor based on his knowledge of the subject. For this user model the only information accessible to a proactive search agent in this environment is the user-written content ( $W_t$ ) and the documents that the user has opened for access in the recent past ( $R_t$ ).

As described in Section 6.2, a proactive agent always keeps track of a user's activities and shows proactive suggestions after fixed time intervals. Since it is a continuous process, we split the simulated environment into different cycles to estimate the effectiveness of a proactive IR model at different instances. Each cycle corresponds to a particular interval of user actions where it starts with user activities and ends with a proactive suggestion list shown to the user. The simulation setup

used in our research scope was motivated by the work in (Koskela et al., 2018). This study simulated a user’s writing activity using an article on a topic. The results obtained from the simulation model correlated with a user study where a user was asked to write an article on a topic. Similar to (Koskela et al., 2018), we also simulate a user’s reading and writing activity using a set of documents related to a topic. Next we describe the simulated read and write cycles in detail.

### 6.3.1 Simulated Reading Cycles

For reading cycles, we focus on reading activities only. In the simulation setup, the user is initialized with a subset of documents (i.e.  $R_0$ ) that they have accessed previously. A user will only interact with documents for a certain amount of time if he finds the document useful for an information need. Hence in our simulated environment, constructing  $R_0$  (i.e. the initial reading set) corresponds to automatically extracting a random subsample from this set of relevant documents for a particular topic. The reason for using random subsampling is that we did not want to create bias in the initial selection of documents on a particular topic. Mathematically  $R_0 \subset \mathcal{R}$  where  $\mathcal{R}$  is the set of relevant documents corresponding to the topic in which the simulated user is currently interested in.

The cardinality of this set  $R_0$  depends on a user’s knowledge. If a user is more knowledgeable, it is more likely that he would have accessed a greater number of relevant documents corresponding to a topic. Hence mathematically we can write  $|R_0| = K|\mathcal{R}|$  where  $K$  is the parameter denoting a user’s knowledge in our simulation setup. The initial Active Set  $A_0$  is initialized with  $R_0$  for simulated reading cycles, mathematically  $A_0 = (R_0, W_0 = \phi)$ . One assumption for constructing  $R_0$  is that, we have access to manually judged relevant documents for a topic considered in the simulation setup.

The behaviour of a user  $U(K)$  is then simulated as follows. As a next step, we assume that the user reads through parts of these documents (i.e.  $R_0$ ) which make it possible for a proactive search agent to get access to the recently read content (a

real-life implementation might make use of an eye tracking device (e.g. Puolamäki et al. (2005)). A relevant document for a topic contains a mixture of sentences where some sentences are directly related to the overall topic of the document, and some are not directly related to the overall topic of the document. As a matter of fact, the content read by different users is likely to be a mixture of relevant and non-relevant sentences from these documents in different proportions, i.e. the eye-tracked contents of different users are likely to be constituted of different numbers of relevant and non-relevant sentences.

To model the output of an eye tracking device while a user is reading a document, in our simulated environment we select a random subsample (without replacement) of relevant sentences and mix it with a random subsample (without replacement) of non-relevant ones as shown in Equation 6.1. For the simulated reading activity generation process described in Equation 6.1, our assumption is that we have the relevance and non-relevance labels of sentences for each relevant document corresponding to a topic. In the absence of relevance and non-relevance labels we can randomly select a set of sentences from a document.

$$R_1 = \bigcup_{\forall d \in R_0} \{(x_r, x_n) \in (r(d), n(d)) : \frac{|x_r|}{|x_n|} = \alpha\}, \quad (6.1)$$

A proactive agent is not aware of the fact that the simulated content has a mixture of relevant and non-relevant sentences and treats all the sentences in the simulated content with equal priority. In Equation 6.1,  $R_1$  represents the read state of the modified Active Set (after the simulated read activity) at the next relative time instant,  $d$  represents a document from  $R_0$ , and  $x_r$  and  $x_n$  represent a relevant or non-relevant sentence respectively from the set of relevant and non-relevant sentences of  $d$ , denoted respectively by  $r(d)$  and  $n(d)$ . The parameter  $\alpha$ , named as *signal to noise ratio* controls the ratio of mixing relevant sentences with non-relevant sentences in a simulated read content.

The proactive agent then uses  $R_1$  (the Active Set constructed after  $R_0$ ) to for-

mulate a query and retrieve a ranked list of documents potentially relevant to the current topic of interest to the user (the topic for which the documents in  $R_0$  are relevant). The top  $M$  retrieved documents are shown as a notification list to the user.

The next step of user simulated interaction involves simulating the user selecting documents from this notification list, similar to a real user who would identify relevant documents from their titles or snippets. The relevant documents newly retrieved by the proactive system are added to the Active Set of documents for the next iterative step as shown in Equation 6.2.

$$R_2 = R_1 \cup \{d_i : d_i \in \mathcal{R}\}_{i=1}^M, \quad (6.2)$$

In Equation 6.2,  $M$  is the number of documents shown in a notification window, and  $\mathcal{R}$  is the set of relevant documents for the particular topic. As already explained in Section 6.2.1, our assumption in the simulation setup is that we have access to all the relevant documents corresponding to a topic.

The simulated user then continues with the read operation with the new Active Set of documents, and the proactive agent also continues to leverage information from the user's reading cognitive state and notifying the user with more documents, i.e., the user and the proactive search agent repeatedly cycle through the steps outlined in Equations 6.1 and 6.2.

Recall from Chapter 3 that in our proactive suggestion framework, proactive search is performed after a fixed number activities. Consequently in our simulation setup, we also trigger proactive suggestion when we have identified a certain number (e.g. 2 or 3) of simulated content items (i.e. read content in this scenario). After the first proactive suggestion, whenever a proactive agent comes up with a new ranked list of documents for proactive suggestion, we preprocess this list by removing any document (i.e. relevant or non-relevant) that has already been shown to user in any of the previous proactive suggestions. The proactive suggestions stop when the



proactive model cannot provide any more new relevant documents corresponding to the user’s current task topic after a search operation for a revised query.

We use random subsampling so that there is no bias (e.g. choosing those relevant documents which are more effective for query formulation compared to others) in choosing documents for a user’s initial state. To make our experiments less biased towards a specific subsample, we execute the simulated steps of interactions on a number of different subsamples (specifically 5), and then report the average results over these runs of the simulated user sessions.

### **6.3.2 Simulated Reading and Writing Cycles**

In contrast to the read-only operation which assumes that a proactive agent is aware of the initial Active Set of documents that a user has accessed recently (e.g. opened for reading), we consider a more general case in which we assume that the user’s knowledge does not explicitly map to a set of relevant documents. In practice, this situation occurs when a user has prior knowledge on a topic, and does not explicitly open a set of files for reading within the operating environment. This is because either he does not need to refer to these documents (his existing knowledge being adequate for his content creation task), or he does not remember if he has saved relevant documents for the writing task and if he does, where to find them. The assumption in this simulation setup is that from the moment the proactive agent starts tracking user activities, the simulated user was involved in writing.

The task of the proactive agent is more challenging in such a situation because unlike the discussion in Section 6.3.1, it has no idea about the documents that the user has read which could be used to formulate a query and retrieve information that the user will find useful. In such a situation, the only content available to a proactive agent is the content generated by the user as a step towards his goal (e.g. writing a research report on a topic on which the user has prior knowledge).

Since natural language generation is a challenging problem in itself, the process of new content generation as a part of the user writing event is difficult to simulate.

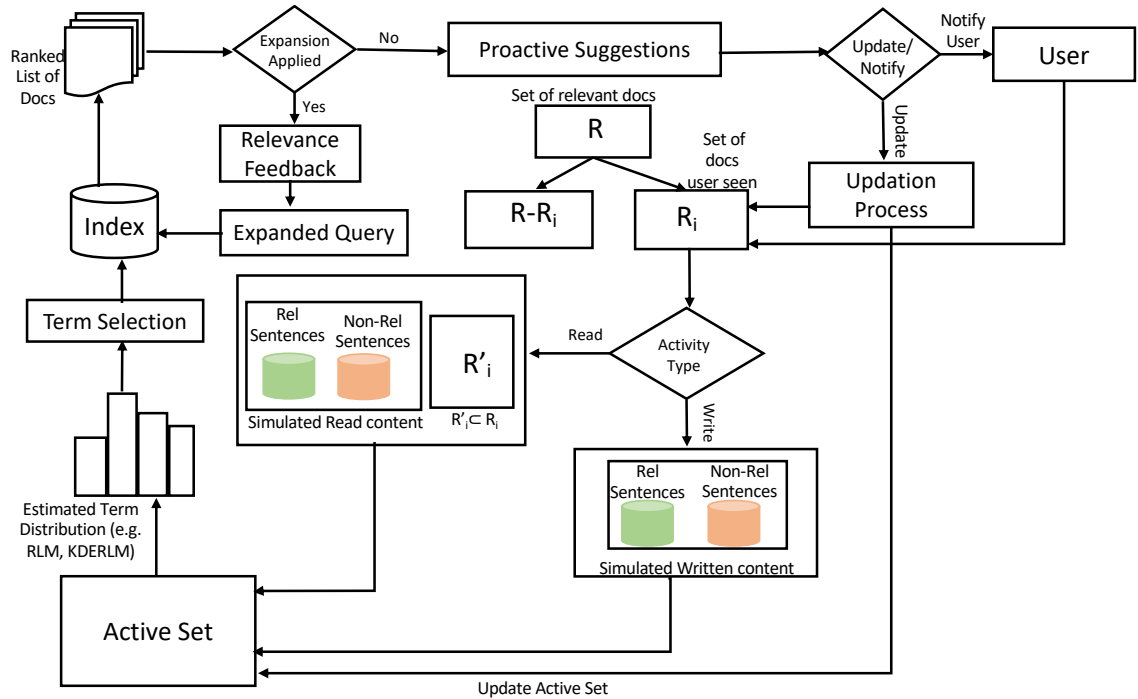


Figure 6.2: Work Flow of Proactive IR Model in Our Proposed Simulated User Setup.

A user written text is generally in the form of a sequence of sentences. Not all the sentences written by a user may be directly related to the overall topic on which the user is writing. Hence as a solution, we adopt a writing simulation process which extracts pieces of text from a mixture of relevant and non-relevant sentences of a subset of relevant documents, similar to the reading simulation step (Equation 6.1). This represents a relatively simple behaviour of a user replicating parts of his acquired knowledge as newly created content. The difference between this and the initial state of the read event cycle is that the proactive agent can only make use of the extracted elements of content placed into the new text for query formulation, and not the entire set of relevant documents  $R_0$  from which the content is created (i.e. the content of which is known only to the user).

The proactive agent then notifies the user with a ranked list of relevant documents, the content of which the user may find useful. The simulation then proceeds with the assumption that the user includes the relevant documents from the retrieved list into his cognitive state, and continues to extract a mixture of relevant

and non-relevant sentences from these additional documents by the process of simulated reading and writing. In our simulation setup, the reading and writing cycle continues until most of the relevant documents corresponding to a topic have been shown as proactive suggestion to the user.

Figure 6.2 shows the workflow of a proactive IR model in our proposed simulation setup. Comparing Figure 6.2 with Figure 6.1, We can see how we simulate the read and written contents of a user, which in a real user setting would be done by tracking the read or write activities of the user by comparing Figure 6.1 with Figure 6.2. It is shown in Figure 6.2, that a simulation user can have two either read or write activities. Figure 6.2 also shows, how the Active Set is initialized differently with read and write activities. For a simulated reading cycle the Active Set has access to both the documents that the user is reading and the simulated output of an eye tracking device. For simulated writing the user has access to only the texts written by the simulated user. Figure 6.2 also illustrates our two options of either showing the ranked list of documents retrieved using the original query or expanding the original query and then using the corresponding ranked list as proactive suggestions to the user. We next describe the query formulation process in our simulation setup in detail.

## 6.4 Query Formulation for Single Stage Task Setup

In this section, we describe how our proactive search agent leverages information from the Active Set of its environment to formulate a query and retrieve documents for a user. In Section 6.2.2, we described how the Active Sets evolve in response to these different user activities in the read-only and read-write cases. Theoretically a proactive search agent is only aware of the information that is made available to it by the user interactions, i.e., our proactive agent can get information only through user actions of: a) opening a document, b) reading parts of it, c) clicks on notifications of retrieved documents, or d) new content creation.

We use the query formulation framework described in Chapter 3. Recall from Equation 3.3 in Chapter 3, we need two components to formulate a proactive query: one is the user's most recent activity (i.e. activity at time  $t$  denoted as  $a_t$  described in Chapter 3), and the background set  $B_t$ . In Chapter 3, we showed that  $B_t$  comprises of a user's recent activity context (i.e.  $C_t$ ) and similar activities of the user himself and other users from their past activities (i.e.  $H_t$ ). In the context of a simulation setup for single stage task,  $C_t$  consists of a series of simulated activities  $a_1, \dots, a_{t-1}$  obtained from the active set at time  $t - 1$  (i.e.  $A_{t-1}$ ) and  $H_t$  consists of only the past activities of the user. Recall from Section 6.2.2, that the past activities of the simulated user is represented by the initial Active Set (i.e.  $A_0$ ).

We do not have information about activities of other users in this individual task scenario. Hence  $H_t$  has only the past activities of user himself. All of the interactions of the user in our simulation environment are based on a single topic rather than on multiple topics. Hence we do not need to find past activities related to a user's current activity (i.e.  $H(a_t)$  as described in Chapter 3). Here all the simulated activities are related to a user's current overall task goal. As a result of this,  $H(a_t)$  is exactly equal to  $H_t$  in our simulation setup. Mathematically  $B_t$  in the simulation setup is equivalent to the active set at time  $t$  (i.e.  $A_t$ ).  $A_t$  is shown in Equation 6.3.

$$A_t = \bigcup_{i=0}^t \{a_i\} \quad (6.3)$$

From Chapter 3, we again describe the query formulation model as shown in Equation 6.4.

$$P(w|\theta_R, a_t) = \prod_{t \in a_t} \sum_{a \in A_t} P(t|a) P(w|a) e^{-\delta(a_t, a)^2} \frac{a_t a}{|a| |a_t|}. \quad (6.4)$$

In Equation 6.4,  $e^{-\delta(a_t, a)^2}$  uses the time difference between an activity  $a$  and the most recent activity  $a_t$ . In our simulation setup we do not have any explicit timestamp associated with each simulated activity. However, we have the information about the sequence of activities. So if an activity was simulated at  $i^{th}$  sequence (i.e.  $a_i$ ) and another activity was simulated at  $j^{th}$  sequence (i.e.  $a_j$ ) then the time

difference between  $a_i$  and  $a_j$  is computed as  $(i - j)$ .

## 6.5 Experimental Setup

In this section, we describe the experimental setup and objectives for our simulation experiments. Recall from Section 6.2 that the objective in this chapter is to simulate a user’s interaction with applications in a desktop environment, and examine the use of a proactive agent to provide support to the user in this simulated environment. In this section, we first describe the specific research questions that we investigate through our experiments, and then we discuss how we use an existing IR dataset (with topics and relevance assessments) to simulate the environment of a user’s interactions.

### 6.5.1 Experiment Objectives

Envisioning the proactive recommendation agent described in Section 6.3 leads to a number of important points to consider. Firstly, how effectively can we exploit the Active set to provide proactive suggestions to the user. Secondly, since different users can have different levels of knowledge on the same topic. The same document may not be useful to all the users for a particular topic. As a result of this the effectiveness of proactive suggestions can vary with a user’s knowledge. Thirdly, we focus on two different types of simulated user activities (i.e. read and read/write) described in Section 6.3.1 and 6.3.2. The initial Active Set of a simulated user in the read and read/write scenarios is not the same. Hence the effectiveness of a proactive suggestion will vary based on the simulated user activities. Based on the above discussions we focus on following three aspects of our research questions in our empirical investigations as follows.

**RQ3-a:** How effectively can we support a user in a simulation setup using proactive query formulation method described in Chapter 3?

Stats	Value
#Total docs (Disks 4,5 - CR)	528,155
#Topics	50
#Avg relevant docs per topic	36.36
#Avg relevant sentences per doc	4.92
#Avg sentences per doc	42.66

Table 6.2: Details of the TREC-2002 Novelty Track dataset used in our experiments on proactive IR.

**RQ3-b:** How does the simulated knowledge of individual users affect the effectiveness of a proactive agent?

**RQ3-c:** How do the effectiveness of proactive suggestions compare across two different types of user activities, namely reading and writing?

The overall objective of the above three research questions (i.e. RQ3-a, RQ3-b, RQ3-c) is to examine whether we can formulate queries from a user’s recent desktop activities and use these queries to provide useful information to the user as described in the research questions in Chapter 1.

### 6.5.2 Dataset

Since our experiments require a set of both manually assessed relevant documents and sentences for a particular topic, we use the TREC Novelty Track dataset for our investigation (Harman, 2002). The TREC Novelty track involved finding a set of relevant and *novel* sentences for a user information need. Details of the dataset are outlined in Table 6.2. The document collection used for this task is the TREC ad-hoc task document collection (disks 4 and 5 without Congressional Records similar to (Roy et al., 2016)). A sentence is judged to be relevant by an assessor if the sentence, in a stand-alone manner, served to address the topic of the information need. A sentence was judged novel if it conveys a new piece of information in a specified order of document presentation. The order of document presentation is fixed for each given topic in the dataset. This modeled the order in which documents would

be presented to a searcher in the output of an IR system. Figure 6.3 shows a sample topic and its corresponding manually judged two relevant sentences from Document ‘LA031689-0177’. As shown in Figure 6.3, each topic has *desc* and *narrative* fields.

For our investigation, we make use of only the relevance information of the documents and sentences, since we do not address the novelty aspect in the scope of this study. Specifically, in our work, each topic corresponds to a different cycle of interactions between a simulated user and the proactive agent. Recall from Section 6.3.1 and Section 6.3.2 that a simulated user starts with an initial knowledge base. The Active Set is initialized with the documents corresponding to this initial knowledge base. For our experiments, a proportion of the set of judged relevant documents for a topic is used as the initial Active Set of an interaction. This fraction is specified by one of the user model parameters,  $K$ . As an example, for a particular instantiation of a user (say with  $K = 0.2$ ),  $1/5$ -th of judged relevant documents are sampled to be used as the initial Active Set  $R_0$ . The proactive agent uses these to suggest the rest of the relevant documents to the user to help him in his current task.

To simulate the content read by a user we randomly select a number of sentences from the document which the user is currently reading. For writing, we assume that a user has previously read a few documents, and written content based on the information gathered from those documents. The simulated written content is generated by randomly sampling sentences from the documents that user has already read. Since we use random sub-sampling, to avoid bias in experiment results we repeat the experiment 5 times and report the results in Tables 6.3, 6.4 and 6.5.

In terms of generating simulated read or written content, we use another parameter  $\alpha$ , which we call the *signal to noise ratio*. This is designed to control the proportion of relevant and non-relevant sentences in the content that has been read by the user or has been written by a user. For the research paper writing goal, this corresponds to what proportions of the user written sentences are related to the core topic in which the user is interested.

For retrieval purpose, we indexed the TREC disks 4,5 (without the Congressional

<num> Number: 305 </num>  
 <title> Most Dangerous Vehicles </title>  
 <desc>Description: Which are the most crashworthy, and least crashworthy, passenger vehicles?</desc>  
 <narr> Narrative: A relevant document will contain information on the crashworthiness of a given vehicle or vehicles that can be used to draw a comparison with other vehicles. The document will have to describe/compare vehicles, not drivers. For instance, it should be expected that vehicles preferred by 16-25 year-olds would be involved in more crashes, because that age group is involved in more crashes. I would view number of fatalities per 100 crashes to be more revealing of a vehicle's crashworthiness than the number of crashes per 100,000 miles, for example. </narr>

Manually judged relevant sentences from document 'LA031689-0177' for topic 305.

- Times Staff Writer The federal government's highway safety watchdog said Wednesday that the Ford Bronco II appears to be involved in more fatal roll-over accidents than other vehicles in its class and that it will seek to determine if the vehicle itself contributes to the accidents.
- 43 Bronco II single-vehicle roll-overs caused fatalities.

Figure 6.3: Sample topic and manually judged relevant sentences for the corresponding topic 305 in TREC Novelty Track Dataset.

Records) collection in Lucene<sup>1</sup> after removing stopwords using the SMART list<sup>2</sup> comprising 571 words and applying Porter stemmer. We used the Stanford core-NLP package<sup>3</sup> to split a document into sentences for the simulation of reading and writing activities.

### 6.5.3 Evaluation Setup

We use the evaluation framework described in Section 4.4 of Chapter 4 to evaluate the effectiveness of our proposed proactive IR model for single stage task scenarios. We proposed two different types of evaluation metrics in Chapter 4:

- One for evaluating the quality of the proposed proactive query .

---

<sup>1</sup><https://lucene.apache.org/>

<sup>2</sup><https://www.lextek.com/manuals/onix/stopwords2.html>

<sup>3</sup><https://stanfordnlp.github.io/CoreNLP/>



- One for evaluating the overall effectiveness of the proactive IR model.

In the single stage task simulation setup, we do not have an original query typed by the user. Hence rather than focusing on measuring the quality of the proactive query (i.e. since we do not have a baseline query for comparison), here we focus only on evaluating the effectiveness of the proactive suggestions of documents retrieved using the proactive query.

Recall from Section 4.4.2 in Chapter 4 that to measure the overall effectiveness of the proposed proactive IR model, we need to compute a reference set at each instance of proactive suggestions. We also explained in Section 4.5.1 of Chapter 4 that for a single stage task simulation setup, the reference set consists of the judged relevant documents corresponding to the topic on which the user is currently interacting with the computing environment.

With every proactive suggestion we update the reference set by removing any relevant documents that have already been suggested by the proactive IR model. Equation 6.5 describes the computation of the reference set mathematically. Using the reference set we compute  $P@5$ ,  $P@10$ ,  $MRR$  and cumulative recall, as reward functions, described in Equation 4.6 in Chapter 4, to measure the overall effectiveness of the proactive IR model.

More specifically, for the simulation setup we report the reward functions (i.e.  $P@5$ ,  $P@10$ ,  $MRR$ , Cumulative Recall) in two different ways:

- One is the weighted average over  $n$  number of proactive suggestions (as described in Equation 4.7 in Chapter 4) starting from the proactivity starting point (i.e.  $\pi$ ). Since in our simulation setup we start suggesting documents from the beginning of the current task, the  $\pi$  value is always 1 in this setup. We weight the reward metrics at a particular proactive suggestion point, based on the number of simulated activities completed before the proactive suggestion at that point. We denote the weighted average of the reward functions corresponding to  $P@5$ ,  $P@10$  and  $MRR$  as  $PE - P@5$ ,  $PE - P@10$ ,  $PE - MRR$

respectively in this chapter. Only cumulative recall is reported as the total recall value at the end of  $n^{th}$  proactive suggestion.

- For further analysis, we also report the reward functions (i.e. P@5, P@10, MRR, cumulative recall) computed at each proactive suggestion pass separately.

Since each reference set is not an ordered set (i.e. the reference is a collection of documents, there is no relevance order among the documents of the set) in a single stage task setup, we cannot compute a correlation coefficient between the predicted ranked list of suggestions and an ideal ranked list obtained from reference set in this scenario.

$$R_t = R - \bigcup_{j=0}^{t-1} \{r_j\}, \quad (6.5)$$

#### 6.5.4 Investigation of different Term Selection Approaches

In our simulation setup, our objective is to formulate a proactive query given an activity context which has a bag of words representation. Existing term selection approaches (e.g. Okapi (Robertson, 1990), RLM (Lavrenko and Croft, 2001) and KDERLM (Roy et al., 2016)) can be used in this scenario in two different ways:

- i) to Formulate a proactive query.
- ii) to expand an already formulated proactive query.

#### Proactive Query Formulation using Existing Term Selection Approaches

We first compare the effectiveness of our proposed query formulation approach (i.e. QFM) with existing term selection approaches. The term selection approaches that we use as our baselines are: Okapi-BM25 (Robertson, 1990), RLM (Lavrenko and Croft, 2001) and KDERLM (Roy et al., 2016). The objective of using these baselines is to observe how our proposed query formulation approach (i.e. QFM) performs

with respect to established term selection approaches in our simulation setup. All the baselines and our proposed approach (i.e. QFM) estimate the importance of a term given a user’s simulated activity context (i.e. simulated read or read/write content). The top  $n_1$  terms are used to formulate the proactive query. The results of this comparison are shown in Table 6.3.

### Query Expansion Using Existing Term Selection Approaches

Here we focus on expanding already formulated proactive queries using the term selection approaches mentioned above. The term selection approaches, namely Okapi, RLM and KDERLM, estimate the importance of an expansion term from feedback documents. Then top  $k_1$  terms are chosen to expand the original query.

In the standard relevance feedback literature, the top  $m$  retrieved documents corresponding to a query are used as a pseudo relevance feedback signal to expand the query. For our query expansion experiments reported in Tables 6.4 and 6.5, we use top  $m$  retrieved documents of the proactive query, formulated in first pass, for pseudo relevance feedback similar to the existing IR literature.

For the subsequent passes, we focus on true relevance feedback, rather than pseudo relevance feedback. Thus, we utilize simulated user clicked documents, rather than top  $m$  retrieved documents as feedback which are then used to expand the query. The reason for considering simulated clicked documents as feedback signal is that existing literature (Li et al., 2017; Yu et al., 2014) shows that clicked documents are likely to be indicative of the documents that user may find useful. For our simulation setup, this is implemented by including only the truly relevant documents from the retrieved list similar to (Chappell and Geva, 2010; Brondwine et al., 2016) for Okapi, RLM and KDERLM.

Our objective is to observe whether query expansion using relevance feedback can improve our proposed query formulation approach or not. The results of applying query expansion terms in a proactive query formulation approach are reported in Tables 6.4 and 6.5. We now turn our attention to outline the details of each of the

term selection approaches examined below.

### Okapi-BM25

Okapi-BM25 (Robertson, 1990) is a standard IR approach to identify important terms from a set of documents predicted to be associated with document relevance. In our simulation setup, we have user generated read and write content and the documents that user has accessed as a source of relevant information corresponding to an information need. The formula for estimating the weight of each term using Okapi BM25 is shown in Equation 6.6. In Equation 6.6  $w(t)$  is the weight for a term  $t$ ,  $N$  is the total number of documents in the collection,  $R$  is the number of relevant documents considered and  $r$  is the number of relevant documents where a term  $t$  occurs and  $n$  is the number of documents where the term  $t$  occurs. Once we obtain  $w(t)$  values, then we compute  $r * w(t)$  to estimate the final weights as suggested in (Robertson, 1990).

$$w(t) = \frac{(r + 0.5)/(R - r + 0.5)}{(n - r + 0.5)/(n - N - R + r + 0.5)} \quad (6.6)$$

### RLM

Recall from Section 2.1.2 in Chapter 2 that RLM is a state-of-the-art approach to IR which uses feedback documents to estimate the importance of a word based on the relevance of that term to the corresponding information need. Similar to the Okapi-BM25 baseline, RLM (Lavrenko and Croft, 2001) can also be applied on relevant documents to estimate the significance of the words with respect to relevance. The formula for computing the weight of a word in RLM is described in Equation 6.7. In Equation 6.7,  $a_1, a_2, \dots, a_k$  are the set of simulated read or write activities of the user.

$$P(w, a_1, a_2, \dots, a_k) = P(w) \sum_{i=1}^k P(a_i|w) \quad (6.7)$$

## KDERLM

Recall from Section 2.1.2 in chapter 2 that the method proposed in (Roy et al., 2016) is also one of the most recent approaches where embedding is used along with RLM method to estimate the relevance of a word with the corresponding information need. The advantage of KDERLM over RLM is that because of the use of word embedding KDERLM addresses semantic similarity between terms along with word co-occurrence to choose the top terms. The formula for estimating the weight of a term in KDERLM is described in Equation 6.8. In Equation 6.8,  $M$  is the collection of simulated activities (i.e.  $a_1, a_2, \dots, a_k$ ) in a user's recent past.

$$P(w|R) = P(w|M) \prod_{i=1}^k P(a_i|M) \quad (6.8)$$

For all the methods including the baseline approaches and the proposed query formulation approach we use BM25 as our retrieval model. The reason to use BM25 is because it has been demonstrated to work well on the IR ad-hoc tasks on standard benchmark datasets (Robertson et al., 1994; Robertson, 1990). The parameters  $k$  and  $b$  for BM25 need to be selected. Figure 6.4 shows the variation of  $P@5$  values with different values  $k$  and  $b$  for all the methods. While varying  $k$  in Figure 6.4 (i.e. left diagram in Figure 6.4), we kept the value of  $b$  as 1. For varying  $b$  we kept the value of  $k$  as 1. Since for all the methods optimum  $P@5$  values were observed for  $k = 1$  and  $b = 1.5$ , for all our methods we used these values of  $k$  and  $b$ . We also use similar configurations of  $k$  and  $b$  in Tables 6.4 and 6.5.

We also varied the number of terms considered for query formulation. Figure 6.5 shows the variation of  $P@5$  values for different values of number of query terms. The optimum value of  $P@5$  is observed when number of query terms is 5. Hence the number of query terms for all the results reported in Table 6.3 are 5.

For KDERLM and QFM, we use the skip-gram model (Mikolov et al., 2013a) trained on TREC 6,7 and 8 to obtain the word vectors. To choose the optimum value for word embedding dimension we show the variations in  $P@5$  values with change in

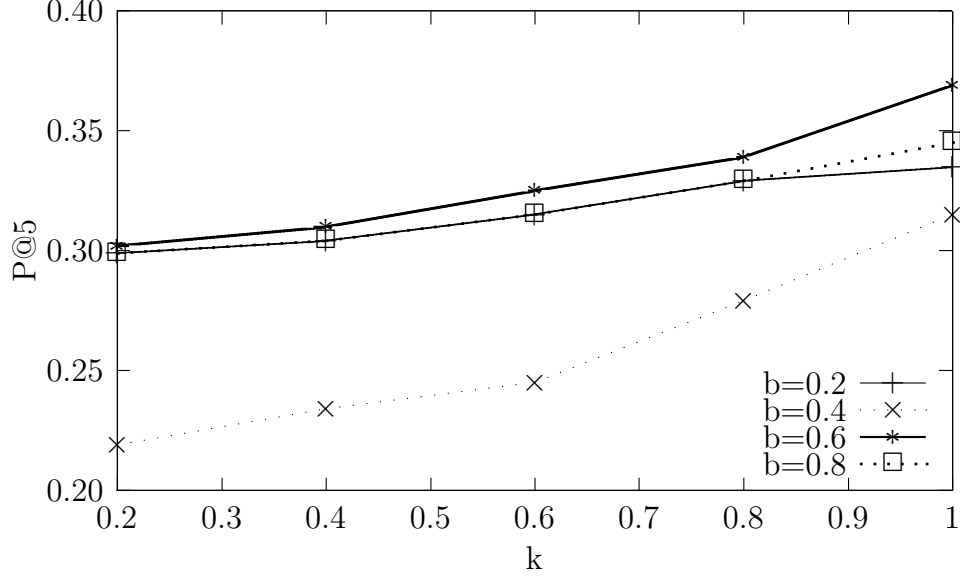


Figure 6.4: Each line in the plot corresponds to a particular value of  $b$  in BM25 parameters. We show the variation in  $P@5$  values with the variation of BM25 parameters  $k$  for Step 1 in RLM. Due to the use of the random construction of the initial knowledge of the user, the results shown in the plot reflect the average value over 5 different runs.

the dimension of word vectors from 100 to 300 in the interval of 100 in Figure 6.6. The  $P@5$  values reported in Figure 6.6 are for step 1 in simulated read activities. It can be observed from Figure 6.6 that the optimum value of  $P@5$  is obtained at dimension 200 for both QFM and KDERLM. As a result of this the results reported in Table 6.3 use word vectors of length 200 for both QFM and KDERLM.

Since we consider this as a high Precision IR task where the user will only typically wish to consult a small number of documents of top ranked retrieved documents, we choose  $p@5$  for optimizing our parameters among all the different metrics (e.g. MRR, cumulative recall) reported in Tables 6.3, 6.4 and 6.5.

### 6.5.5 Query Formulation Components Analysis

Here we focus on investigating the effectiveness of the different components of our query formulation approach (i.e. QFM) described in Equation 6.4 and the variants of QFM with different query expansion techniques (i.e. Okapi, RLM, KDERLM) through an ablation study. The equation in 6.4 has three components.

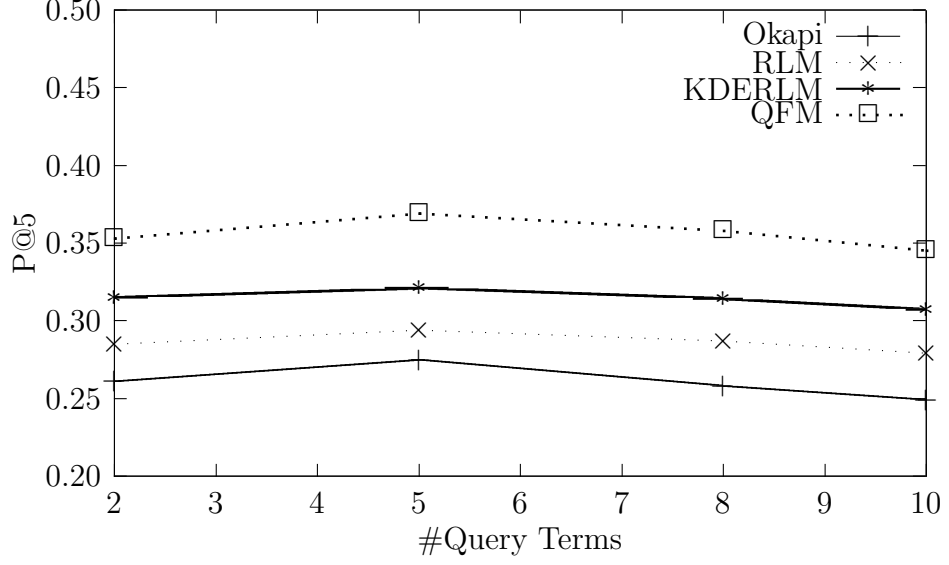


Figure 6.5: Effect of varying the number of query terms in P@5 at step 1 in simulated read activities. The methods considered are Okapi, BM25, RLM, KDERLM and QFM. This plot shows that for all approaches optimum P@5 is observed when the number of query terms were 5.

- The first component computes the co-occurrence of words present in  $B_t$  (i.e. the relevance set is constructed from a user’s most recent and past activities) with  $a_t$  (i.e. user activity at time  $t$ ). In our ablation study, we use the notation ‘CO’ to describe this co-occurrence component in the rest of the chapter.
- The second component is a time decaying factor (i.e.  $e^{-\delta(a_t, a)^2}$ ). In our ablation study, we use the notation ‘TD’ for this time decaying component in the rest of the chapter.
- The third component is the semantic similarity factor ( i.e.  $\frac{a_t a}{|a||a_t|}$ ). In our ablation study, we use the notation ‘Sim’ for this component in the rest of this chapter.

Now we turn our attention to the specific implementation details of our experiments.

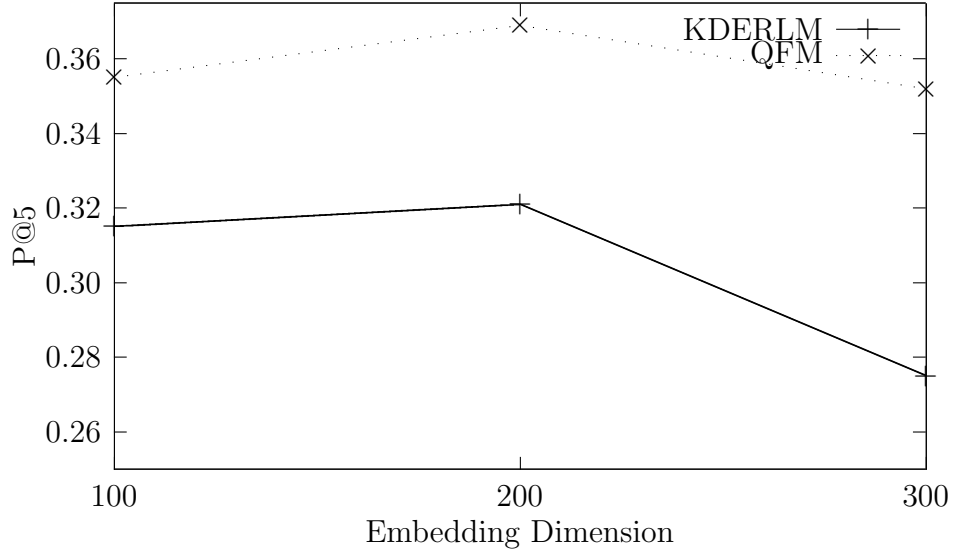


Figure 6.6: Effect of varying the word embedding dimension in QFM and KDERLM for simulated read activities in step 1.

### 6.5.6 Implementation Details

For our simulation experiment, we consider three cycles (i.e. three different proactive suggestion passes described in Section 6.3) of proactive suggestions for each topic in the TREC Novelty Track dataset, as described in Section 6.5.2. We formulate a new proactive query at each proactive suggestion pass. At each pass, we preprocess the proactive suggestion list by removing any relevant or non-relevant documents that have already been shown to the simulated user in a previous proactive suggestion list (i.e. top 10 documents in a ranked list of retrieved documents).

For the relevance feedback experiments (i.e. QFM+Okapi, QFM+RLM, and QFM+KDERLM), we need to choose the number of expansion terms to the original query. Figure 6.7 shows the variation in  $P@5$  values with change in the number of expansion terms for methods, namely, QFM+Okapi, QFM+RLM, and QFM+KDERLM. For all the methods, the optimum value is observed when the number of expansion terms were 3. As a result of this, for all the query expansion experiments reported in Tables 6.4 and 6.5 we add 3 expansion terms to the original query.

Recall from Section 6.3.1 and 6.3.2 that to simulate read and write activities



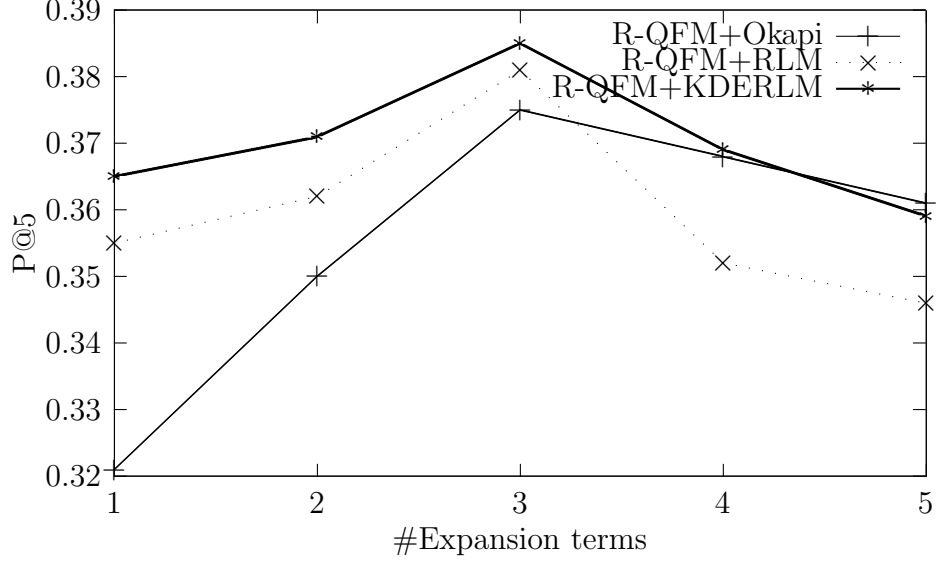


Figure 6.7: Effect of varying the number of expansion terms in P@5 at step 1 in simulated read activities. The methods considered are QFM+Okapi, QFM+RLM, QFM+KDERLM.

we sub-sample a mixture of relevant and non-relevant sentences from the Active Set. The total number of sentences considered in generating a simulated read/write activity is set to 5 for all our experiments.

In our experiments, we retrieve the top 10 documents (in accordance with existing work which reported that 10 is the optimal size for a search result page (Kelly and Azzopardi, 2015b)). Passing notifications to real-life users with more than 10 documents may cause long interruptions in the user’s current reading or writing activities. Thus, we assume users will not be willing to inspect more than 10 suggested documents.

## 6.6 Results of Simulated Proactive IR

In this section, we present the results of our investigation of proactive query suggestion models in our simulation setup. We first describe the effectiveness of our proposed query formulation technique (i.e. QFM) compared to existing term selection approaches. Then we examine use of query expansion techniques applied on QFM and the ablation of different components in QFM and its query expansion

Simulation Type	Model	$K$	$\alpha$	Evaluation Metric			
				PE-MRR	PE-P@5	PE-P@10	Cum-Recall
Read	Okapi	0.4	0.5	0.209	0.110	0.105	0.341
Read	RLM	0.4	0.5	0.228	0.121	0.110	0.372
Read	KDERLM	0.4	0.5	0.241	0.133	0.121	0.396
Read	QFM	0.4	0.5	<b>0.255<sup>†</sup></b>	<b>0.152<sup>†</sup></b>	<b>0.132<sup>†</sup></b>	<b>0.472<sup>†</sup></b>
Read-Write	Okapi	0.4	0.5	0.203	0.106	0.101	0.361
Read-Write	RLM	0.4	0.5	0.218	0.123	0.189	0.391
Read-Write	KDERLM	0.4	0.5	0.239	0.133	0.205	0.398
Read-Write	QFM	0.4	0.5	<b>0.274<sup>†</sup></b>	<b>0.144<sup>†</sup></b>	<b>0.125<sup>†</sup></b>	<b>0.403<sup>†</sup></b>

Table 6.3: Comparison of our proposed query formulation approach (i.e. QFM) with existing term selection approaches for simulated read and read-write activities. For all the query formulation approaches BM25 is used as a retrieval model and the number of query terms is 5. The ‘<sup>†</sup>’ symbol indicates statistical significance (paired t-test with 95% confidence) of QFM in comparison to the best baseline approach (i.e. KDERLM), e.g. the value  $PE-MRR = 0.255$  in QFM (read) is significantly better than  $PE-MRR = 0.241$  in KDERLM (read).

versions.

### 6.6.1 QFM Performance Compared to Baselines

For all the methods in Table 6.3, simulated users were initialized with same knowledge (i.e. the initial number of relevant documents accessed by the user were same across all the methods) and no query expansion was applied on all the methods reported in Table 6.3. The metrics reported in Table 6.3 are the weighted (i.e. weighted in terms of the number of simulated activities required to provide the proactive suggestion) average over three proactive suggestions in simulated setup. Table 6.3 shows that our proposed query formulation approach (i.e. QFM) outperforms existing state of the art term selection approaches in standard IR (i.e. Okapi, RLM, KDERLM) for both read and read-write activities (i.e. First and second half of the Table 6.3) in terms of all the evaluation metrics.

Figures 6.8, 6.9 and 6.10 show the variation of MRR and P@5 and Cumulative Recall across the three different proactive suggestion points in the baseline approaches and QFM. It is clearly visible from Figure 6.10 that the P@5, MRR and cumulative recall values of QFM are greater than or equal to each of the baselines at

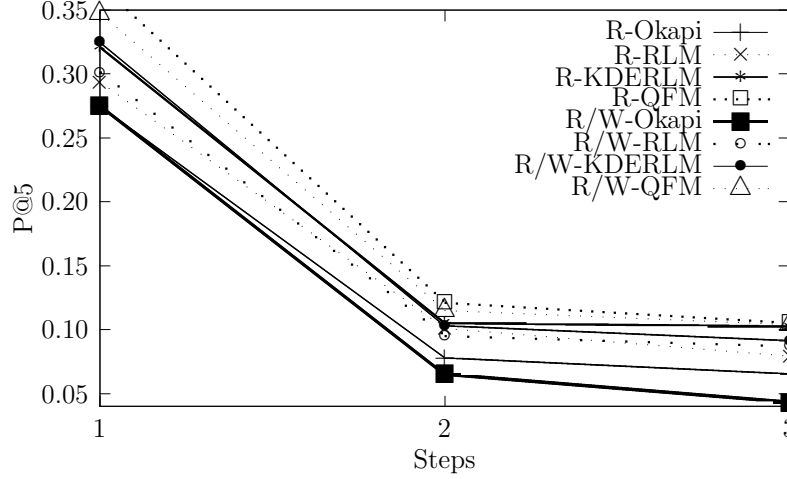


Figure 6.8: P@5 at different steps of proactive suggestion in our simulation setup. ‘R’ denotes the read activity and ‘R/W’ denotes the read-write activity in this Figure.

each proactive suggestion points.

It is also observed from Figures 6.8, 6.9 and 6.10 that for all the methods the value of MRR and P@5 decrease with the progression in steps and cumulative recall is increasing as expected. The decrease in P@5 or MRR values are less from step 1 to step 2 compared to step 2 to step 3 (i.e the slope of the straight line computed between consecutive steps). One potential reason can be that with the progress in time it becomes difficult to bring new relevant documents corresponding to the same information need.

We wanted to investigate for how many passes we should provide proactive suggestion to the simulated user. Hence we varied the number of passes and observed the values of MRR, P@5 and cumulative recall with the progression of steps for simulated read activity in QFM in Figure 6.11. It can be observed from Figure 6.11 that with the increase in step, the values of  $P@5$  and MRR are decreasing to 0 and the value of cumulative recall is saturating. In the fifth step the values of  $P@5$  and  $MRR$  are almost zero and the value of cumulative recall is almost same as step 4. The values of P@5 and MRR at step 4 are in the order of  $10^{-2}$ . From this observation, we can say that it becomes difficult to bring new relevant documents beyond step 3. Similar trend is observed for all other baseline approaches and for simulated

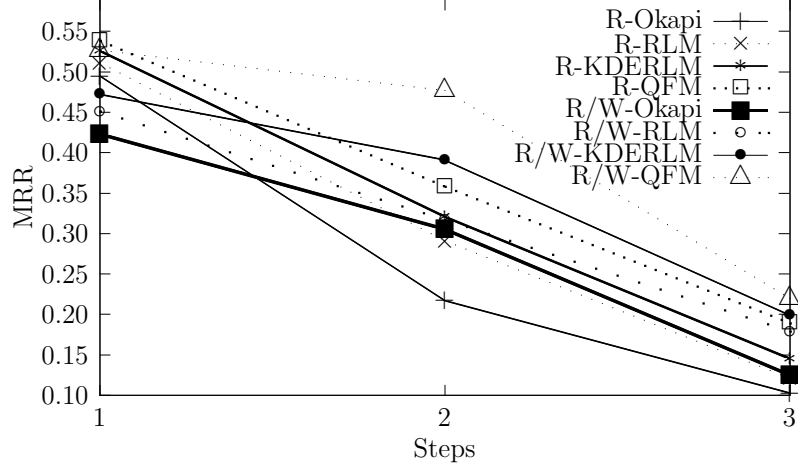


Figure 6.9: MRR at different steps of proactive suggestion in our simulation setup. ‘R’ denotes the read activity and ‘R/W’ denotes the read-write activity in this Figure.

write activities. As a result of this, the total number of proactive suggestion passes for which we reported the results in all of our experiments in Table 6.3, 6.4 and 6.5 is 3.

### 6.6.2 Ablation Results on Simulated Read and Read-Write Activities

Tables 6.4 presents the ablation study results of proactive search effectiveness(averaged over 50 topics present in TREC Novelty track) obtained with simulated read-only interactions (i.e. where the simulated user only reads a portion of relevant documents) for users with a fixed initial knowledge. Specifically, we take  $K = 0.4$  portion of relevant documents as a starting point of reading simulation. All the results reported in Table 6.4 are averaged over 5 different simulation runs. Each method in the ablation study, namely QFM, QFM+Okapi, QFM+RLM and QFM+KDERLM, has either one, two or no components absent. Based on the presence and absence of a component in a method it is denoted by ✓ and ✗ respectively in Tables 6.4 and 6.5. For example in a method described as (CO(✓) TD(✗) SIM(✗)), the component CO is present and the components TD and SIM are absent. We report all possible combinations (i.e. 7 different types) for each category of method in our experiments. We

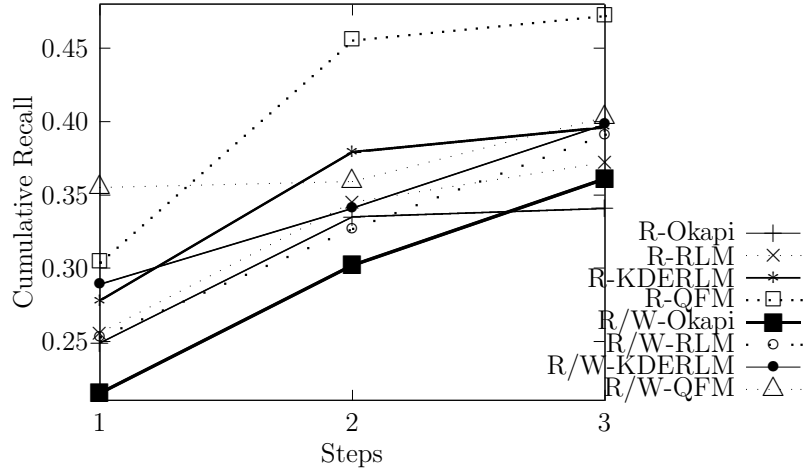


Figure 6.10: Cumulative Recall at different steps of proactive suggestion in our simulation setup. ‘R’ denotes the read activity and ‘R/W’ denotes the read-write activity in this Figure.

do not consider the method where none of the components are present (i.e.  $\times \times \times$ ) since without any of the components it is not possible to formulate a query using our proposed query formulation method (i.e. QFM).

The observations from Table 6.4 are as follows.

1. The ablation study in terms of the three components (i.e. CO, TD and SIM) shows that for all the different types of methods with and without query expansion (i.e. QFM, QFM+Okapi, QFM+RLM, QFM+KDERLM), including all the three components performs the best in terms of all the evaluation metrics.
2. Among the first three rows for all the ablation study categories (i.e. QFM, QFM+Okapi, QFM+RLM and QFM+KDERLM) in Table 6.4, the first row performs the best in terms of all the evaluation metrics. Hence we can say that among all the three components (i.e. CO, TD and SIM) ,CO is the most important component in our proposed query formulation approach since
3. The combination of our proposed approach and query expansion using KDERLM (i.e. 28th rows in Table 6.4 mostly outperforms BM25 and RLM in terms of all the evaluation metrics (i.e. P@5, P@10, MRR, Cumulative Recall).

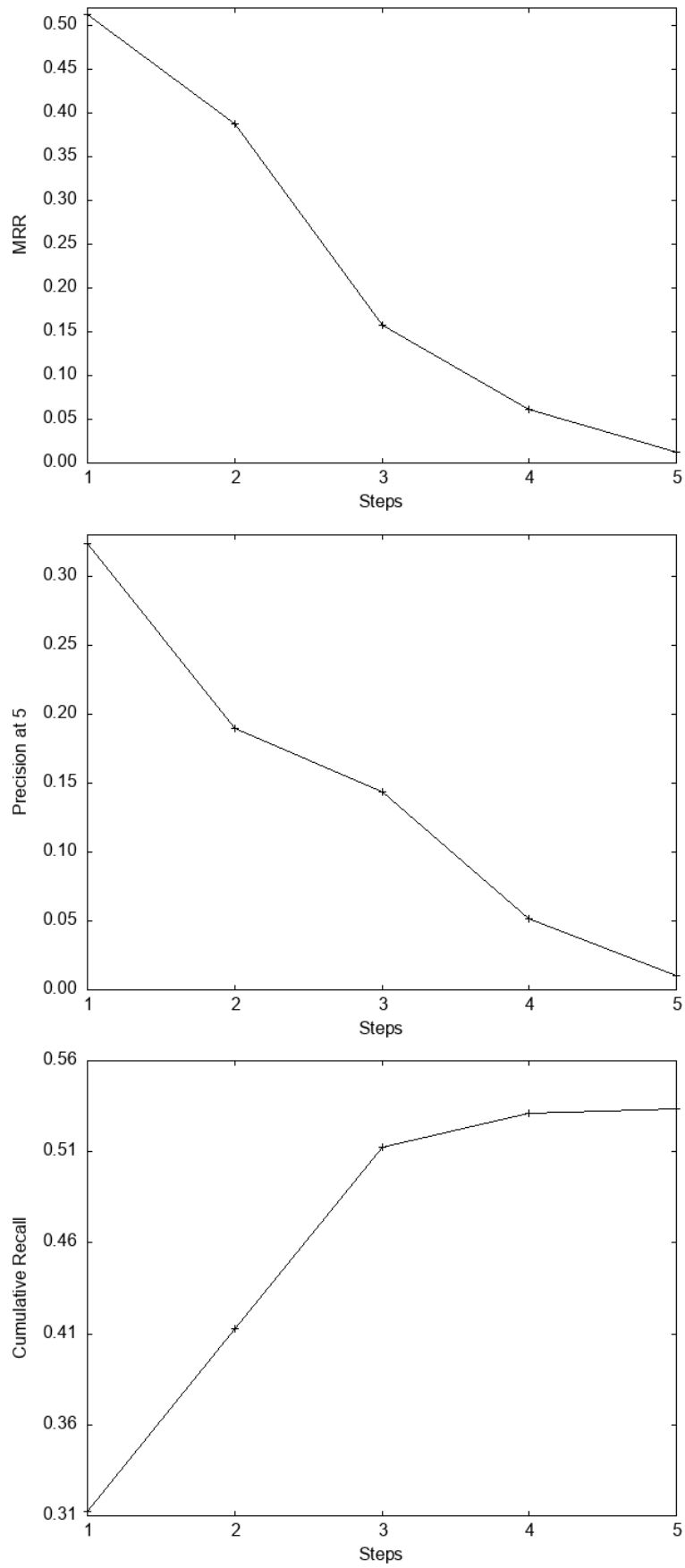


Figure 6.11: Variations in the values of MRR, P@5 and Cumulative Recall with time progression (i.e. increase in steps) in simulated setup.

Table 6.4: Results of proactive IR effectiveness with simulated read activities of user agents.

Model	CO	TD	SIM	Expansion	$k$	$\alpha$	Evaluation Metrics			
							PE-MRR	PE-P@5	PE-P@10	Cum-Recall
QFM	✓	✗	✗	No	0.4	0.5	0.237	0.127	0.099	0.385
	✗	✓	✗	No	0.4	0.5	0.192	0.101	0.095	0.370
	✗	✗	✓	No	0.4	0.5	0.195	0.116	0.089	0.381
	✗	✓	✓	No	0.4	0.5	0.220	0.134	0.095	0.390
	✓	✗	✓	No	0.4	0.5	0.246	0.145	0.106	0.423
	✓	✓	✗	No	0.4	0.5	0.251	0.147	0.115	0.465
	✓	✓	✓	No	0.4	0.5	<b>0.255</b>	<b>0.152</b>	<b>0.132</b>	<b>0.472</b>
QFM + Okapi	✓	✗	✗	Okapi	0.4	0.5	0.241	0.136	0.100	0.390
	✗	✓	✗	Okapi	0.4	0.5	0.231	0.117	0.125	0.345
	✗	✗	✓	Okapi	0.4	0.5	0.234	0.116	0.129	0.330
	✗	✓	✓	Okapi	0.4	0.5	0.240	0.129	0.155	0.412
	✓	✗	✓	Okapi	0.4	0.5	0.249	0.137	0.106	0.422
	✓	✓	✗	Okapi	0.4	0.5	0.257	0.143	0.150	0.475
	✓	✓	✓	Okapi	0.4	0.5	<b>0.267</b>	<b>0.153</b>	<b>0.165</b>	<b>0.489</b>
QFM + RLM	✓	✗	✗	RLM	0.4	0.5	0.246	0.145	0.126	0.451
	✗	✓	✗	RLM	0.4	0.5	0.239	0.137	0.118	0.432
	✗	✗	✓	RLM	0.4	0.5	0.243	0.134	0.115	0.441
	✗	✓	✓	RLM	0.4	0.5	0.246	0.143	0.120	0.448
	✓	✗	✓	RLM	0.4	0.5	0.251	0.146	0.131	0.469
	✓	✓	✗	RLM	0.4	0.5	0.264	0.149	0.162	0.531
	✓	✓	✓	RLM	0.4	0.5	<b>0.280</b>	<b>0.157</b>	<b>0.172</b>	<b>0.544</b>
QFM + KDERLM	✓	✗	✗	KDERLM	0.4	0.5	0.252	0.142	0.158	0.459
	✗	✓	✗	KDERLM	0.4	0.5	0.243	0.132	0.126	0.432
	✗	✗	✓	KDERLM	0.4	0.5	0.238	0.126	0.150	0.441
	✗	✓	✓	KDERLM	0.4	0.5	0.245	0.141	0.148	0.443
	✓	✗	✓	KDERLM	0.4	0.5	0.252	0.147	0.160	0.471
	✓	✓	✗	KDERLM	0.4	0.5	0.270	0.150	0.168	0.544
	✓	✓	✓	KDERLM	0.4	0.5	<b>0.279</b>	<b>0.159</b>	<b>0.176</b>	<b>0.573</b>

Table 6.5 presents the weighted average of the proactive evaluation metrics (i.e. PE-MRR, PE-P@5, PE-P@10 and Cumulative Recall) over three steps for simulated read-write interactions for users (i.e. where the simulated user perform both read and write activities) for QFM, QFM+Okapi, QFM+RLM and QFM+KDERLM. The observations in Table 6.5 are similar to those of Table 6.4.

However, in relation to our research question RQ3-C (i.e. How proactive suggestion effectiveness varies across two simulated read and read-write activities?), it can be said that in terms of all the evaluation metrics, the values of  $P@5$ , MRR and cumulative recall of write initiated interactions are lower than that of read initiated interactions in Table 6.4. One possible reason for this may be that for read-write

Table 6.5: Results of proactive IR effectiveness with simulated read/write activities of user agents.

Method	CO	TD	SIM	Expansion	$k$	$\alpha$	Evaluation Metrics			
							PE-MRR	PE-P@5	PE-P@10	Cum-Recall
QFM	✓	✗	✗	No	0.4	0.6	0.218	0.119	0.101	0.338
	✗	✓	✗	No	0.4	0.6	0.202	0.109	0.082	0.321
	✗	✗	✓	No	0.4	0.6	0.186	0.099	0.091	0.315
	✗	✓	✓	No	0.4	0.6	0.209	0.106	0.088	0.331
	✓	✗	✓	No	0.4	0.6	0.209	0.121	0.111	0.356
	✓	✓	✗	No	0.4	0.6	0.253	0.126	0.116	0.387
	✓	✓	✓	No	0.4	0.6	<b>0.274</b>	<b>0.144</b>	<b>0.125</b>	<b>0.403</b>
QFM + Okapi	✓	✗	✗	Okapi	0.4	0.6	0.220	0.120	0.118	0.356
	✗	✓	✗	Okapi	0.4	0.6	0.207	0.109	0.105	0.341
	✗	✗	✓	Okapi	0.4	0.6	0.200	0.109	0.098	0.326
	✗	✓	✓	Okapi	0.4	0.6	0.211	0.112	0.108	0.349
	✓	✗	✓	Okapi	0.4	0.6	0.218	0.132	0.147	0.361
	✓	✓	✗	Okapi	0.4	0.6	0.239	0.140	0.152	0.416
	✓	✓	✓	Okapi	0.4	0.6	<b>0.255</b>	<b>0.148</b>	<b>0.162</b>	<b>0.431</b>
QFM + RLM	✓	✗	✗	RLM	0.4	0.6	0.239	0.121	0.138	0.377
	✗	✓	✗	RLM	0.4	0.6	0.230	0.101	0.130	0.358
	✗	✗	✓	RLM	0.4	0.6	0.221	0.095	0.127	0.342
	✗	✓	✓	RLM	0.4	0.6	0.228	0.097	0.132	0.349
	✓	✗	✓	RLM	0.4	0.6	0.245	0.108	0.139	0.367
	✓	✓	✗	RLM	0.4	0.6	0.266	0.134	0.169	0.429
	✓	✓	✓	RLM	0.4	0.6	<b>0.283</b>	<b>0.150</b>	<b>0.185</b>	<b>0.436</b>
QFM + KDERLM	✓	✗	✗	KDERLM	0.4	0.6	0.250	0.120	0.138	0.436
	✗	✓	✗	KDERLM	0.4	0.6	0.240	0.107	0.130	0.418
	✗	✗	✓	KDERLM	0.4	0.6	0.224	0.099	0.125	0.498
	✗	✓	✓	KDERLM	0.4	0.6	0.232	0.111	0.134	0.429
	✓	✗	✓	KDERLM	0.4	0.6	0.249	0.138	0.147	0.461
	✓	✓	✗	KDERLM	0.4	0.6	0.279	0.146	0.150	0.492
	✓	✓	✓	KDERLM	0.4	0.6	<b>0.285</b>	<b>0.152</b>	<b>0.166</b>	<b>0.516</b>

activities, the initial information available to the proactive search agent is less compared to read only activities. For read-write activities the Active Set for step 1 is limited by the knowledge of the new content written by the users. In this case, the system has no way of knowing the content of the documents on which the written content was based on (i.e.  $B_t$  in Equation 6.4 has only the simulated written content it does not have the original document). Whereas for simulated read activities, the proactive search agent has information about the content read by the user along with the whole document that the user was reading (i.e.  $B_t$  in Equation 6.4 has both the simulated read content and the content read by the user).

Figure 6.12, 6.13 and 6.14 show the value of reward functions (i.e. MRR, P@5,



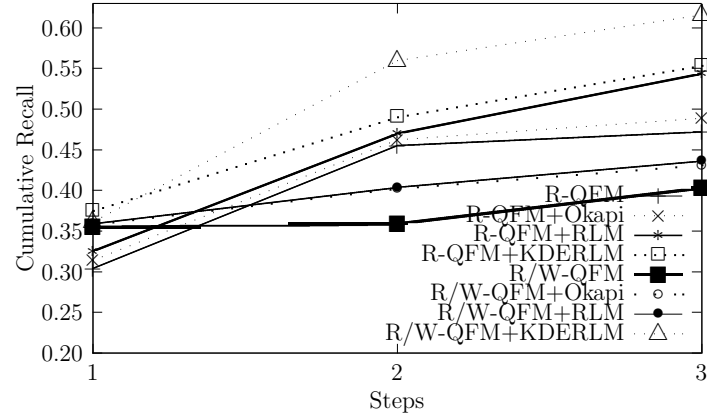


Figure 6.12: Variations in MRR in Different Steps for QFM and the different variations of QFM using query expansion techniques (i.e. QFM, QFM+Okapi, QFM+RLM and QFM+KDERLM).

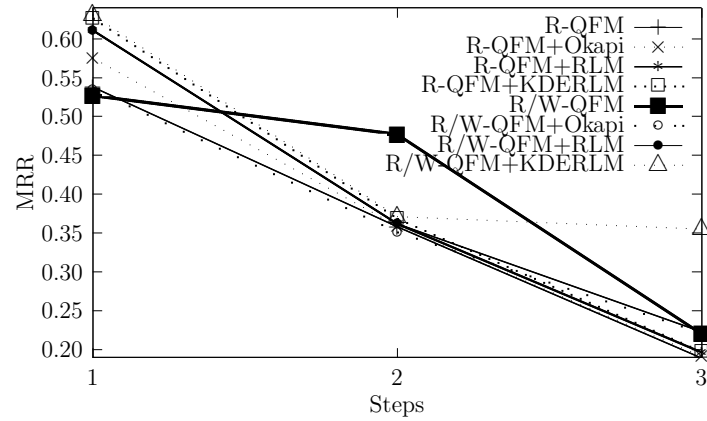


Figure 6.13: Variations in P@5 in Different Steps for QFM and the different variations of QFM using query expansion techniques (i.e. QFM, QFM+Okapi, QFM+RLM and QFM+KDERLM).

Cumulative Recall) at different proactive suggestion points for QFM, QFM+Okapi, QFM+RLM, and QFM+KDERLM for both simulated read and write activities. It can be observed from Figure 6.14 that for the subsequent iterative steps after the first (i.e. second and third iterations) cumulative recall tends to saturate (i.e. top left of Figure 6.14) and P@5 values (i.e. bottom of Figure 6.14) tend to decrease, which suggests that it is difficult to retrieve previously unseen relevant documents for successive steps. This situation is particularly true for simulated user behaviour because the simulation environment can only make use of a mixture of relevant and non-relevant sentences from either the seed set of relevant documents or the relevant documents retrieved during successive stages.

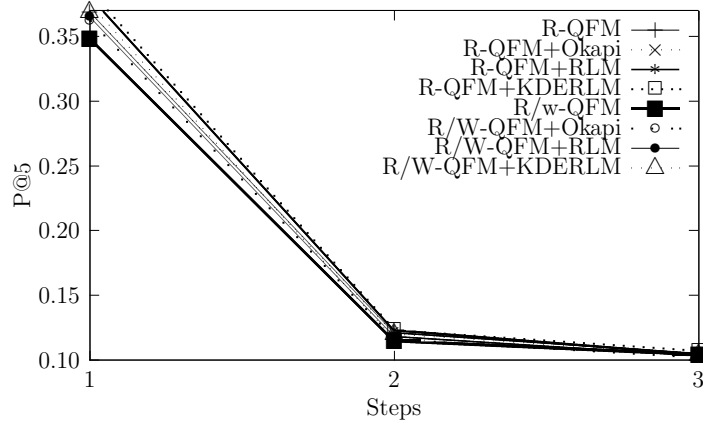


Figure 6.14: Variations Cumulative Recall in Different Steps for QFM and the different variations of QFM using query expansion techniques (i.e. QFM, QFM+Okapi, QFM+RLM and QFM+KDERLM).

In Figure 6.17, for step 2 we add two different versions of results for  $P@5$ : The first one corresponds to the scenario where we use the simulated clicked documents by the users as feedback signal (i.e. true relevance feedback) for query expansion using Okapi, RLM or KDERLM, we name this step2a. The second one corresponds to that scenario where we apply feedback for query expansion two times, we name this step2b. The reason for demonstrating both step2a and step 2b is to observe whether pseudo relevance feedback applied after a true relevance feedback works better for proactive suggestion or not. It can be observed that the results in the step 2b column are lower than the step 2a column in most cases. So we did not compute the other evaluation metrics (i.e.  $P@10$ , MRR and Cumulative Recall) for the step 2b scenario in Figure 6.14. For all the other evaluation metrics (i.e.  $P@10$ , MRR, Cumulative Recall) step 2 corresponds to the scenario of step2a in  $P@5$ .

### 6.6.3 Parameter Sensitivity Analysis

In relation to RQ3-b, Figure 6.15 shows the variations of  $P@5$  and MRR values with respect to change in a user's knowledge for our proposed query formulation approach (i.e. QFM) and QFM with other query expansion approaches (i.e. QFM+Okapi, QFM+RLM and QFM+KDERLM) in simulated read activities. Similar analysis is shown in Figure 6.16 for simulated read-write activities. To have a fair compari-

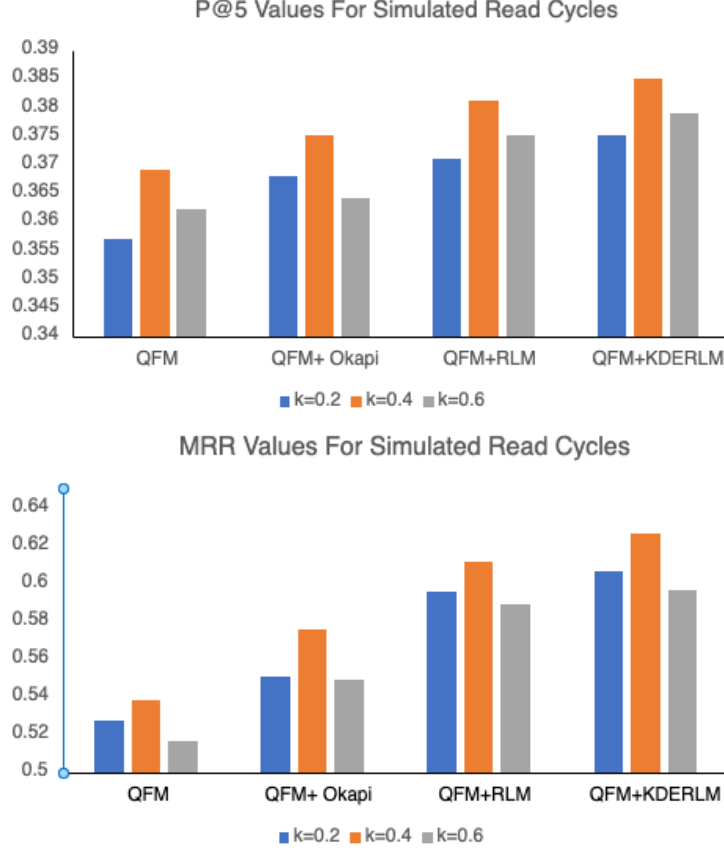


Figure 6.15: Effect of varying the  $k$  in P@5(Top Diagram) and MRR (Bottom Diagram) values for simulated write cycles in Step 1.

son between different knowledge configurations (i.e. the initial number of relevant documents that is used to start simulation for read and read-write activities), we keep the target set of relevant documents that a proactive retrieval model tries to retrieve fixed across different knowledge configurations. It can be observed that more knowledge of a topic (higher value of  $K$ ) does not necessarily help in proactive retrieval. One potential reason can be that starting with a higher number of relevant documents may bias the retrieval, which will try to bring relevant documents similar to the one that user have already seen. But our goal in the simulation setup is to retrieve new relevant documents on the same information need. This is also shown by the lower values of MRR achieved with higher values of  $k$ , e.g. the best MRR with RLM for  $k = 0.2$  is 0.611, whereas the best MRR with  $k = 0.6$  is 0.574. This implies that a proactive search agent is likely to be more effective in finding unknown relevant documents if a user starts an exploration task with a relatively

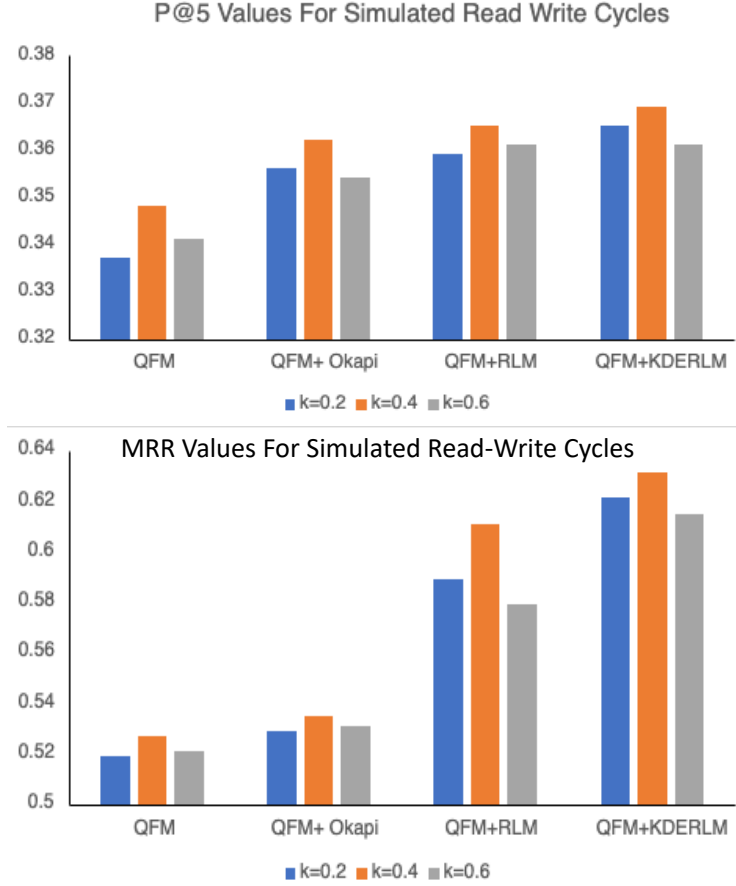


Figure 6.16: Effect of varying the  $k$  in P@5(Top Diagram) and MRR (Bottom Diagram) values for simulated read write cycles in Step 1

small amount of knowledge.

### Effect of $\alpha$ on Proactive Suggestion Effectiveness

We now take a closer look into the effect of varying the parameter  $\alpha$  for simulated write activities. As described in Section 6.3.2, while generating simulated read/write content for a user we use a parameter named  $\alpha$ , which determines the proportion of relevant sentences in a content read/written by a simulated user. The parameter  $\alpha$  is varied from 0 to 1 in the interval of 0.1. Figure 6.18, 6.19 and 6.20 shows the effect of varying  $\alpha$  in terms of P@5, MRR and cumulative recall for different query formulation approaches (i.e. QFM, QFM+Okapi, QFM+RLM and QFM+KDERLM). The best results are obtained at  $\alpha = 0.6$  for all the three metrics mentioned in a simulated write activity. For simulated read activities the optimum results for all

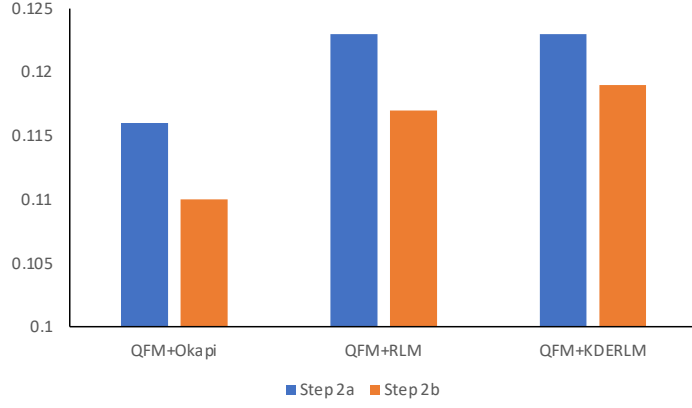


Figure 6.17: Effect of applying true relevance feedback (i.e. step 2a) and true relevance feedback along with pseudo relevance feedback (i.e. step 2b) in the second pass of simulated read activity for QFM

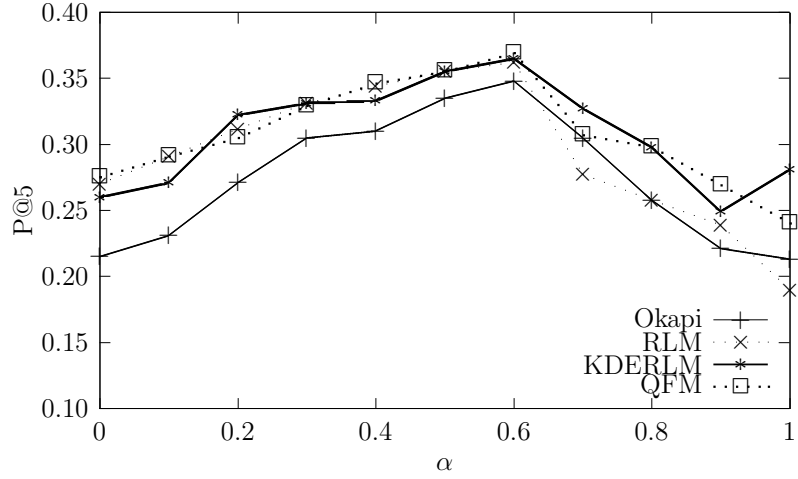


Figure 6.18: Effect of varying the mixing parameter  $\alpha$  ( i.e. used to create simulated read and write content) on P@5 in simulated read activities at step 1.

the evaluation metrics (i.e. p@5, MRR, Cumulative Recall) are observed for a value of 0.5.

## 6.7 Conclusions

In this Chapter, we addressed our third research question RQ3 (i.e. how can we formulate proactive query in single stage task scenario?). We used the query formulation framework introduced in Chapter 3 in a single stage task setup in this chapter.

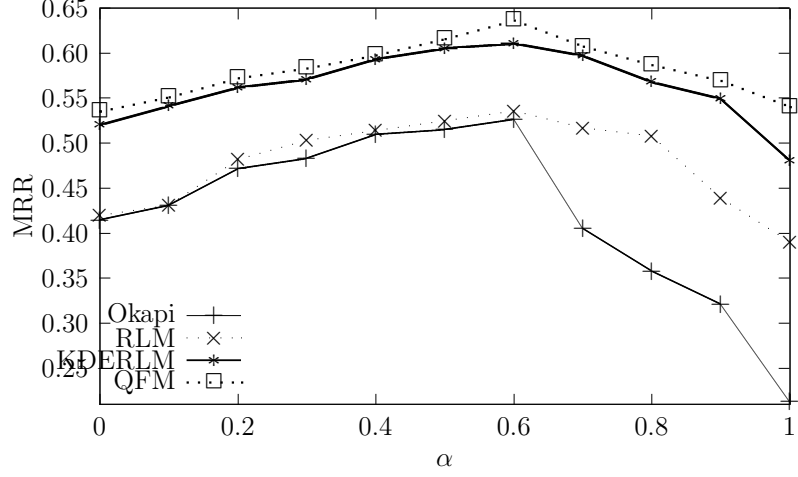


Figure 6.19: Effect of varying the mixing parameter  $\alpha$  ( i.e. used to create simulated read and write content) on MRR in simulated read activities at step 1.

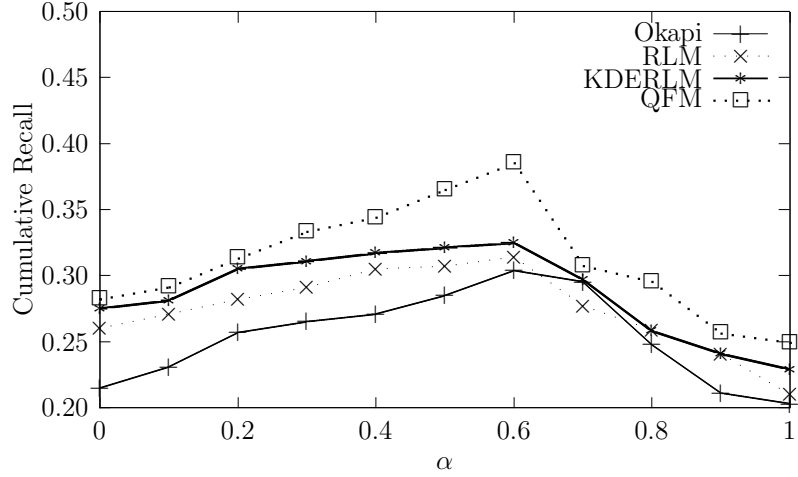


Figure 6.20: Effect of varying the mixing parameter  $\alpha$  ( i.e. used to create simulated read and write content) on cumulative recall in simulated read activities at step 1.

Here we first proposed a laboratory-based reproducible framework for investigating how proactive search agents can interact with a real user (which in this study is simulated). The proactive search agents are designed to track of a user’s recent activities (e.g. read and write) to formulate queries (without user involvement in this process) and to notify users of potentially relevant new documents (ones which the user has not seen before), which they may find useful in carrying out their current task.

In Section 6.5.1 we described the three parts of our RQ3 in the context of simulation setup. In terms of RQ3-a we can conclude from Table 6.3 that compared to

existing unsupervised term selection approaches our proposed method (i.e. QFM) performs better in terms of providing proactive suggestions to the user. In terms of RQ3-b we can conclude from Figures 6.16 and 6.15 that it is not always the case that an increase in knowledge improves proactive suggestion. Beyond a certain threshold if a user has more knowledge then it becomes difficult for a PIR model to provide new relevant documents to the user. In terms of RQ3-c we can conclude from Table 6.3 that given the same configuration our proposed query formulation model is more effective in a read-write scenario compared to a read only scenario.

The observations of our simulated setup may help a practitioner to make design choices in development of an effective proactive IR systems in a setups involving real users, e.g., a) a retrieval model should consider historical interactions of the recent past, in addition to the present context; b) past information should be considered with a decaying factor, etc. We next investigate how same query formulation framework performs for multi stage task setting in Chapter 7.

## Chapter 7

# Proactive Suggestions For Multi-Stage Tasks

In this chapter we address our fourth research question RQ4 ‘*Can we proactively support a user by leveraging similar activities of other users?*’. Here we focus on user tasks where we can leverage similar activities of other users to proactively support the current user. Broadly speaking these are generally tasks where other users have engaged in similar tasks to the one being carried out by the current user. One example of this kind of task might be planning for a vacation to a popular destination. In this case, the different sub-tasks related to the task are booking flights, booking accommodation, knowing about popular places to visit at the destination, etc. A proactive system with access to this information could seek to identify related tasks which have been carried out by other users, and to make use of information extracted from them to support the activities of the current user.

There is no existing publicly available dataset containing a collection of diverse logged activities from multiple users. However, there are publicly available search query logs which contain information about the search activities (i.e. user typed queries, clicked documents) of different users. So we focus here only on the scenario where users are engaged in search tasks to address our fourth research question. While a user is engaged in a search task within a search session, our objective is



to provide proactive suggestions to the user by not only leveraging the user’s past queries (i.e. analogous to past user activities), but also making use of similar search queries typed by other users in the past (i.e. analogous to similar activities being carried out by other users). We described in Chapter 1 that the objective of a PIR model is to support the user in his overall task. To accomplish this objective in our search session setup we anticipate the different information needs that are related to a user’s current task given their current query and provide them with documents related to those information needs to help him complete his task. In a thread of work related to query result diversification (Vieira et al., 2011; Santos et al., 2010), given a user query the objective is to provide diverse documents related to the information need presented in the query. A major difference between our work and this work on query result diversification is that rather than focusing only on the current query context (i.e. query reformulation sequence present in current session), we consider both a user’s long and short term search context to anticipate their overall search task and provide documents related to the possible sub-tasks for the current task of the user.

One limitation of considering only search tasks is that they are controlled in nature (i.e. the user explicitly expresses their information need in terms of search queries). The activities in a search task are only keyword based queries and clicked documents. In the case of general user tasks, activities can be of many different types (e.g. reading, writing documents) and a user may not always have an information need while carrying out an activity. However, if using similar queries from other users can help a user’s current search task, then we believe that similar activities of other users can also be used to help in accomplishing a user’s current task more generally.

In this chapter we first describe how proactive suggestion works during a search session using the framework described in Chapter 3, and then describe our experimental setup used to explore the effectiveness of our proposed proactive suggestion approach using the evaluation framework described in Chapter 4.

## 7.1 Proactive Suggestion in Search Sessions

In this section, we first describe how a user interacts during search tasks and then discuss methods in existing literature that have been proposed to support a user with information sources during search sessions. Our goal is both to review existing work and show how our proposed proactive suggestion method differs from this existing research in supporting users during search tasks.

While interacting with a search system, an appropriate representation of an underlying information need of a user depends not only on the search expertise of the individual user but also on the complexity of the search task, e.g. in a search task with amorphous goals, an information seeker’s knowledge about the overall topic of the task can evolve over time as they interact with retrieved content, making query formulation difficult during different time instants (e.g. first query, middle of search session, end of search session) of the search session (Liu et al., 2016). A traditional IR system necessarily waits for a user to enter a query before computing a ranked list of documents which seek to satisfy their information need, such a system is most likely to satisfy the user’s evolving information needs if the query created at each stage of the information seeking process is relatively well-formed.

To assist inexperienced users or users who are unfamiliar with a topic, standard techniques to make IR systems more proactive include methods such as *query completion* and *query suggestion*. Query completion involves suggestions for completing the remaining part of a partially typed query, e.g. (Mitra et al., 2014). If the original query is ‘web de’, then possible suggestions for query completion might be ‘web design’, and ‘web development’. Query suggestion involves suggesting a list of queries that might better describe the information need and could help to retrieve more relevant documents. These may be related either to the current information need based on previous entered queries similar to the one currently being entered (Feild and Allan, 2013; Li et al., 2012) (*task-agnostic*), or to different directions in which the current information need could evolve (Hassan Awadallah et al., 2014;

Muntean et al., 2013) (*task-aware*).

Both query completion and query suggestion output a ranked list of candidate queries to the user, who can then select a reformulated query from this list, which thus means that these approaches are not completely proactive. Contrastingly, the output of our proposed proactive suggestion approach is a *system-anticipated* ranked list of documents, created entirely automatically which the user may find helpful during the rest of his search session. Such an approach is more proactive than query completion or suggestion, because it does not require the user to take any action interrupting his work.

To retrieve an anticipatory list of documents that might be helpful to the user during the rest of his search session, we use similar queries from other users, taken from a past search log. The objective of this anticipatory list of documents is to show information related to possible information needs associated with sub-tasks of a user's current search activities. Figure 7.1 shows a concrete example of how other users' similar interactions can be used for proactive suggestion. Figure 7.1 is a small excerpt from the AOL search query log introduced in Section 5.5 of Chapter 5. In this example it can be seen that both users  $u_1$  and  $u_2$  have similar information needs. So when user  $u_2$  has typed his first query, we could reasonably use the similar query entered by user  $u_1$  to predict that user  $u_2$  may be interested in knowing about 'robert franci kennedy' after 'john f kennedy family'. Generally speaking, in a traditional search system users like  $u_1$  and  $u_2$  need to manually type initial queries on specific sub-topics (e.g., 'jfk', 'robert f kennedy'), read a number of top ranked retrieved documents to eventually reformulate this query to move on to a different sub-topic (e.g., Kennedy family) during their search session. Our proposed proactive suggestion approach seeks to assist a searcher by retrieving documents related to 'robert francis kennedy' or 'jfk assassination', i.e. by proactively predicting potential subsequent information needs based on the search activities of previous related searches [when a user has only typed a query about 'jfk']. The proactive suggestions are thus likely to reduce the number of user queries within the search

Sessions (AOL User ids)	Queries	
$U_1$ (636926)	jfk robert f kennedy	Similar Information needs of different users
$U_2$ (3458083)	john f kennedy family robert francis kennedy	
$U_3$ (1604008)	jfk assassination	
$U_4$ (3271790)	president john f kennedy assasination crime scene	

Figure 7.1: Sample queries from AOL query log.

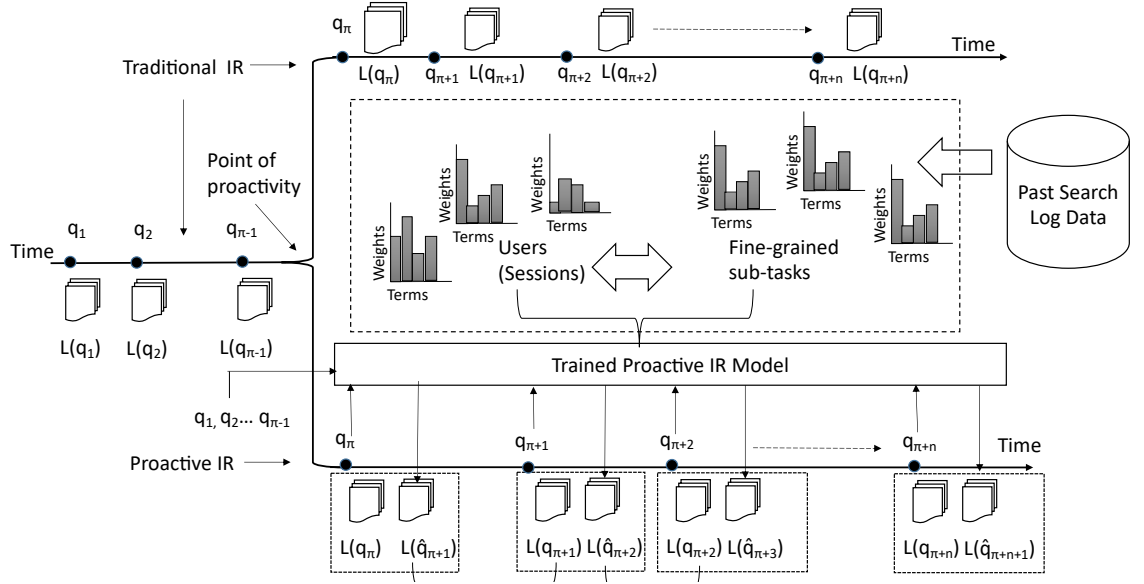


Figure 7.2: Schematic Diagram of a specific proactive search system (PSS).

system (Muntean et al., 2013), and thus the amount of work they need to do.

In the following section, we describe the components and functionality of the architecture of our proposed proactive suggestion model in detail.

### 7.1.1 Overall Architecture

We use the generic framework proposed in Chapter 3 to develop a proactive IR model for search tasks. Figure 7.2 shows the schematic diagram of our proactive suggestion of documents framework in the context of search tasks. We call this framework a proactive search system (PSS). As described in Chapter 3, a PSS essentially needs to use a context of queries previously entered by a user, before it can start retrieving documents without queries being explicitly submitted to the system by the current

user (Tan et al., 2006). Consequently, until the point in time at which a PSS has accumulated sufficient context, it behaves like a traditional search system (as seen in the left part of Figure 7.2), i.e., it retrieves a list of document  $L(q_t)$  in response to a manually submitted user query  $q_t$ , where  $t$  denotes a point in time.

Generally speaking, at some point in time (say at  $t = \pi$ ) within a user search session (marked as ‘point of proactivity’ in Figure 7.2), a PSS can switch to proactive mode. The lower part of Figure 7.2 towards the right of the branch in time  $t = \pi$  shows that in proactive mode, in addition to retrieving a ranked list of documents for the current query submitted by the user,  $q_\pi$ , the system also retrieves a list of documents  $L(\hat{q}_{\pi+1})$ . The objective of the PSS is to seek a list,  $L(\hat{q}_{\pi+1})$ , containing documents which the user may find of interest to accomplish his overall task objective.

We emphasize that it is not the focus of this chapter to investigate if this is the best way to provide proactive suggestions to a user, which is a separate research question to investigate and would rather involve real user studies. Rather the objective of this chapter is to develop a model that could effectively retrieve such a list, which can be used to support a user.

## 7.2 Proactive Query Formulation

As shown in Figure 7.2 in Section 7.1.1, we use three different information sources to proactively formulate a query. They are:

- a user’s recent session context (i.e.  $C_t$  the set of queries from  $q_1$  to  $q_t$ )
- similar queries from a user’s past search sessions (i.e.  $H_{u_t}$ )
- similar queries from other users’ past search sessions (i.e.  $H_{out}$ )

We denote the union of the sets  $H_{u_t}$  and  $H_{out}$  as  $H_t$ . In our query formulation model we give equal importance to all the similar queries coming from the past search log (i.e. the user’s own past search sessions and other user’s past search sessions). In

Section 7.4.2 of this chapter we demonstrate the effect of varying the importance of the user's own search history to other user's search histories.

We use the query formulation model proposed in Section 3.2.1 in Chapter 3 to formulate a proactive query in our search session setup. More specifically, we use Equation 3.7 from Chapter 3 to estimate the weights of different words given the context of a sequence of queries executed by the user.  $B_t$  in Equation 3.7 has two components. One is a user's recent activities (i.e.  $C_t$ ) and the other one is similar activities from the the past (i.e.  $H_t$ ). For convenience we again restate the query formulation method in Equation 7.1.

$$P(w|\theta_R, a_t) = \prod_{t \in a_t} \sum_{a \in B_t} P_e(t|a) P_e(w|a) e^{-\delta(a_t, a)^2} \frac{\mathbf{a}_t \cdot \mathbf{a}}{|\mathbf{a}| |\mathbf{a}_t|}. \quad (7.1)$$

The activities (i.e.  $a'_i$ s) shown in Equation 7.1 are queries in a search session scenario since we consider only search tasks. Recall from Section 3.2.2 in Chapter 3 that we find similar activities related to a user's current activity using an embedding framework. In the last part of the Equation 7.1 we also use embedding based similarity (i.e.  $\frac{a_t \cdot a}{|a| |a_t|}$ ) to estimate the likelihood of a word for formulating a proactive query (i.e.  $p(w|\theta_R, a_t)$ ). We use the graph-based embedding approach proposed in Chapter 5 to obtain the embeddings for both words and queries. In the following section we describe in detail how we obtain the embeddings of both queries and words simultaneously using the graph based approach proposed in Chapter 5.

### 7.2.1 Graph Representation learning

As discussed in Chapter 5, that a graph based embedding approach can help to capture the task semantics better compared to existing approaches. Hence we first construct a graph using queries and words present in the past query log. The nodes of the graph are words and queries present in the log. There are three different kinds of edges in the graph, each of these is described below.

- **Word-Word:** An edge  $(v(w), v(w'))$  exists between a pair of words  $w$  and  $w'$

if they both occur at least once within a search session. If a word pair occurs frequently in multiple sessions across the overall query log, then it is likely that the words in the pair are likely to be associated with the same task topic since all the queries appearing within a single session are generally related to the same task in most of the cases.

- **Query-Query:** Queries appearing within a single search session are likely to be related to a similar topic. Hence we add an edge  $(v(q), v(q'))$  between a pair of queries  $q$  and  $q'$  if they appear at least once within a search session.
- **Query-Word:** An edge  $(v(q), v(w))$  exists between a word and a query (and vice-versa) if the word is a part of that query. The reason for using this type of edge is that a word within a query is related to the overall search intent expressed using the query.

### Defining Edge Weights

The weights of the edges are computed using Equation 5.11 described in Chapter 5. This computes the co-occurrence between a pair of nodes (i.e. word pair, query pair, word-query pair) within a session. For word-word and query-query edges, the weight is given by computing an average of the co-occurrence likelihoods across all sessions. For edges between a query and a word (and vice-versa), we accumulate the occurrence likelihoods as shown in Equation 7.2. In Equation 7.2,  $f(w, q)$  denotes the number of times the word  $w$  is present in the query  $q$  and  $|q|$  denotes the length of the query  $q$ . Essentially  $f(w, q)$  measures the likelihood of word  $w$  appearing in query  $q$ . The higher the value of  $\omega(q, w)$ , the greater the semantic connection between the word and the query.

$$\omega(q, w) = \frac{1}{|S|} \sum_{s \in S} \frac{f(w, q)}{|q|} \quad (7.2)$$

To illustrate the query flow graph with an example, consider the graph (comprising words and queries as nodes) shown in Figure 7.3, constructed using our method





## Sequences vs. Sets

Existing work on query prediction (Sordoni et al., 2015; Chen et al., 2011) uses sequence to sequence modeling to learn query and session representations. But rather than using a sequence-to-sequence model we focus on a graph-based embedding approach for next query prediction. It is worth noting that the main difference in the working principle of a graph-based representation learning of queries and words, with that of a sequence-to-sequence modeling based query suggestion, is that while the latter models the next query as a function of the previously occurring queries within a session, the former is devoid of this assumption. Using a graph-based approach is more suitable for modeling queries seeking to complete a task, because the sub-tasks within a session are mostly independent and could be carried out without any strict reference ordering. A user may proceed to complete his overall task in a different sequence of sub-tasks as compared to his predecessors (i.e. other users who performed similar tasks in the past). A sequence-to-sequence model takes  $q_1$  to  $q_{t-1}$  as input to predict the next query  $q_t$ . Our model is trained with all the queries appearing in the training set of AOL query log data. A sequence-to-sequence model based query suggestion, because of its hard constraints on the strict order observed in the training set (log history), may not generalize adequately. For our proposed approach we first split the whole query log dataset into training and test sets. We use the training data to learn our proposed graph based embedding model. We predict proactive suggestions for a user in the search sessions in the test set. For a given query in the test set, we bring related queries from the training set to provide proactive suggestions of documents to the user.

## 7.3 Experimental Setup

In this section we describe details of our experimental study on proactive search for a multi-stage search task. We begin with an overview of our datasets, the objectives of our experiments, and also describe the application of our Proactive IR evaluation

framework to this task.

### 7.3.1 Dataset

Here we describe how we use an existing search query log data for our experiments for proactive suggestion.

#### Documents and Queries

For this study we again make use of the publicly available AOL query log. In our experiments, we retrieve proactive suggestions from the Clueweb12B Document collection <sup>1</sup>. Clueweb12B contains a total of nearly 52M documents. From this collection, we index only those documents that yield a spam filtering percentile score higher than 70% (Zamani and Croft, 2017) using the spam filtering method proposed in (Cormack et al., 2011). This results in a total of over 13.7M indexed documents from the filtered collection. The combination of the AOL query log and Clueweb12B was previously used in experiments reported in (Zamani and Croft, 2017; Dehghani et al., 2017). In spite of being independent data sources, since the AOL log comprises queries directed to the AOL web search engine, and Clueweb12B is a large general collection of web pages, it is expected that there should be considerable topical overlap between the former and the latter.

#### Defining Sessions

Recall from Section 7.2.1 that our proactive search method relies on the notion of a session to compute co-occurrences of query-pairs or word pairs. We segment the total collection of queries in the AOL log into non-overlapping groups of queries (each group constituting an individual session). We use the same method used in Chapter 5 to split the AOL query log into sessions. A session delimiter was placed whenever the timestamp difference between two consecutive queries exceeded a threshold value (which was set to 26 minutes, as prescribed in (Lucchese et al., 2013)).

---

<sup>1</sup><https://lemurproject.org/clueweb12/>

## Training and Test Sets

The proactive IR model described in Section 7.2 makes use of query logs as a source of potential similar queries entered by other users to proactively support the current user (see Figure 7.2). We split the set of AOL query logs into two parts. One is used for training purposes and the other one is used for testing. Specifically, we define the first 95% (i.e first 20 million queries from AOL log) of the AOL queries as the training set. The queries in the training set are used to form a graph as described in Section 7.2.1, which is used to find queries similar to those of the current user. The queries in the test set are used to evaluate the effectiveness of our proposed proactive query formulation approach, described in Section 7.2.

Recall from Section 7.2 that we need a set of search sessions for our proposed proactive document suggestion model to work. The test split of the AOL query log contains the remaining 1 million queries. To construct the test set from these remaining queries, we first segment these queries into sessions based on the criteria defined in Section 7.3.1. Then from the set of all sessions in the test split, we first select a random subset of 300,000 sessions. Existing literature (Hassan Awadallah et al., 2014) showed that if sessions are longer in length than a session containing two or three queries (e.g. 5 or more), then it is more likely that the user can be engaged in a multi-stage search session. Hence to ensure that the sessions in the test set are representative of multi-stage search sessions (i.e. where the overall search task has multiple sub-tasks in it), we only include those sessions comprising the top 5 percentile (in terms of the session length, or in other words, the number of queries in a session). By top 5 percentile we refer to the fact that we first sort all the sessions in terms of their length in descending order, and if  $x$  is the maximum session length, then we consider only the sessions having length at least  $x * .05$ . This filtering process keeps only those search sessions which are comparatively longer in length. This resulted in a total of 280,567 sessions in the test set. Table 7.1 shows details of the training and the test splits. The number of unique users in the training and test set are 646,009 and 55,000 respectively. The average length of

queries appearing in the training and test set are 5.25 and 5.32 respectively.

## Preprocessing

As a preprocessing step, we first remove stop words from the query log using the SMART stopword list smart (2019). We also remove identical pairs of consecutive queries from the AOL log, since duplicate queries will not provide any extra information. We did not remove identical or similar queries from different sessions. For our graph-based representation learning framework, we first construct a graph from the training set. Since there are 20 million queries in the training set, the constructed graph has a large number of nodes. Hence we employ a frequency threshold to only create nodes for words/queries with collection frequency higher than this threshold across 20 million queries to reduce the size of graph, which will make the computation faster. We examined threshold values from 1 to 5 in the interval of 1. We found that the graph constructed from the threshold value 5 had reasonable size, where graph construction and embedding can be completed in a feasible time. The resulting graph had 1 million nodes and 92,447 edges. Without any filtering the size of the original graph had 1.3 million nodes and 1 million edges.

Moreover, we also employ a threshold to remove spurious edges, where edge weights are below a threshold. If a pair of words or queries co-occur in a very less number of sessions then the corresponding edge weights are low. As a result of this, it is very unlikely that they will carry any useful connection information between a pair of nodes in the graph. Before applying the threshold, we normalized each edge weight within the range  $[0, 1]$  (see Section 7.2.1). We initially varied the threshold parameter from 0 to 0.3 in the interval of .01 and observed the effectiveness of the graph based embedding for proactive suggestion in terms of  $AP$  (i.e. described in the evaluation metric in Chapter 3). The optimum value of  $AP$  was observed for the threshold value of .05.

	Sessions		Queries		#Users
	#Total	Avg Len	#Total	Avg Len	
Training	2,312,177	9.05	16,512,216	5.25	646,099
Test	280,567	19.9	1,578,951	5.32	55,000

Table 7.1: Training and Test splits of the AOL Query log

### 7.3.2 Evaluation Outline

We use the evaluation framework proposed in Chapter 4 to measure the effectiveness of our PSS. As described in Chapter 4, there are two different categories of evaluation metrics for proactive IR. The first one is for measuring the quality of the proactive query, and the second is for measuring the effectiveness of the proactive suggestion list. In Section 4.4.2, we proposed MRR for evaluating query effectiveness. In the search session setup, at time  $t = i$ , we seek to predict a user’s next information need. From the AOL query log dataset we already know the user’s actual next query (i.e. the query formulated at  $t = i + 1$ ). So we can evaluate the proactive query against the user typed query  $q_{i+1}$ . We name this evaluation metric  $QP - MRR$  in our search task scenario.

Recall from Chapter 4 that for each proactive suggestion we need a reference set of retrieved documents for the original query to measure its effectiveness. In our search session setup this reference set is the top  $k$  documents retrieved using the next query actually typed by the user. Concretely speaking, each anticipatory (predicted) ranked list,  $L(\hat{q}_{\pi+i})$ , is compared with the reference list  $L(q_{\pi+i+1})$  in order to measure how successfully a proactive IR model can retrieve the set of documents that would have been retrieved if the user had submitted the query  $q_{\pi+1}$  explicitly. If  $r_t$  denotes an overlap measure between the predicted list of documents  $L(\hat{q}_t)$  and the groundtruth set of documents  $L(q_t)$  at time  $t$ , then an aggregate of these overlaps from the point of proactivity to the end of the current user session estimates the overall effectiveness of a PSS throughout its active duration in a user search session.

Once the reference set is fixed, Section 4.4.2 introduced an alternative reward

function that can be used to compute the overlap between a reference set and a predicted suggestion list. We use MRR (i.e. PREVAL-MRR), precision (i.e. PREVAL-AP) and cumulative recall (i.e. PREVAL- $Ap^+$ ) as described in Section 4.4.2 to measure the effectiveness of our proposed proactive IR model.

### 7.3.3 Approaches Investigated

Here we describe the different approaches that we use as our baselines. A number of existing approaches have been introduced related to TREC Session Track (Carterette et al., 2014) where the objective is to provide a user with all the task related documents based on a given sequence of task related queries executed by a user within a session. A major difference between our work and the methods proposed in (Carterette et al., 2014) is that there is no scope for using similar task related queries of other users in this scenario. Another difference is that in the TREC session track all the task related queries were already given and the objective is to find documents related to these queries. However the objective of our proactive search model is to provide the user with different information needs that could potentially arise in future given a partial context of queries in a search session. Broadly speaking, we employ two categories of baselines, one that uses text-only information and the other that uses representation learning (embedding), each one of them is described in more detail below. For this task, the objective of using text-only baselines is to show how conventional text-based methods perform compared to embedding based approaches, which seek to capture semantic similarity between a pair of queries or words for proactive suggestion. The reason for using a number of embedding based baselines is to observe how existing embedding approaches compare to text-based methods and to compare them to our proposed embedding approach for proactive query suggestion.

## Text-only Baselines

In these baselines we use the word overlap to compute the similarity between a pair of words or queries (i.e.  $\sigma$  in Equation 7.1 in Section 7.2). An objective of using different types of text-based approaches as baselines is to observe how effective word-based similarity for proactive suggestions is compared to approaches which use embedding methods to compute similarity between a pair of queries or terms.

**Relevance Model (RLM) (Lavrenko and Croft, 2001).** This baseline examines a text-only based approach by applying the relevance model (i.e. introduced in Chapter 2). The objective of using this approach as one of our baselines is to observe how using only a user’s most recent query activity (i.e.  $q_t$ ), we can proactively suggest queries to the user for his future information needs. Specifically, we apply the relevance model on the retrieved list of  $q_t$  to estimate a distribution of words based on the user’s current information need. Then we choose top 5 words from the set of words to predict the next query  $\hat{q}_{t+1}$ . We use  $\hat{q}_{t+1}$  to retrieve a ranked list of document suggestions for the user. This baseline neither uses information from the user’s history (i.e.  $H$ ) nor a user’s recently typed queries within a session (i.e.  $C_t$ ).

**Session-Based Relevance Model (SRLM).** The objective of using this approach as one of our baselines is to observe how using a user’s most recent query activity history (i.e.  $q_1, q_2, \dots, q_t$ ), we can proactively suggest queries to the user for his future information needs. In this baseline we apply the relevance model (Lavrenko and Croft, 2001) on the set of queries that the user has typed up to time  $t$  to estimate the importance of a word given a user’s current session context. Based on the weight estimated by the SRLM, we choose the top  $k$  terms to predict  $q_{t+1}$ . The SRLM uses the same working principle as RLM. The formula for weight estimation using the SRLM is given in Equation 7.3. In Equation 7.3,  $q_t$  is the last query typed by a user and  $q_i$  is any previous query appearing in the current session up to time  $t - 1$ . It does not use the information from a user’s past history (i.e.  $H$ ).

$$P(w|R) = P(w|q_t) \prod_{i=1}^{t-1} P(w|q_i) \quad (7.3)$$

**Contextual RLM (CRLM).** The objective of using this baseline is to observe how proactive suggestion works when exploiting a user’s recent query history (i.e.  $C_t$ ) and similar queries from the past ( $H_t$ ), while using only text-based similarity. This baseline is exactly the same as our proposed proactive query formulation approach described in Equation 7.1. The only difference is the  $\frac{a.a_t}{|a||a_t|}$  component of Equation 7.1 uses the embedding of activities/queries to compute the similarity between them. But we use word-based similarity between a pair of queries for query formulation purposes in CRLM. In particular for our experiments, we use an index constructed from the AOL queries from the training set and retrieve the top  $k$  queries from those retrieved based on LM-JM similarity between a pair of queries. To select the subset of candidate queries we set the parameter  $k = 30$ .

**Query Flow Graph (QFG).** In our proposed proactive suggestion framework an existing query prediction model can be used as an intermediate step, where the predicted next query can be used to retrieve a ranked list of documents that can be shown as proactive suggestions to the user. The objective of using this QFG baseline is to observe how an existing graph based query prediction model performs for proactive suggestion.

Here we use a query prediction approach named Query Flow Graph (Boldi et al., 2008; Bonchi et al., 2012) to first predict a candidate set of the  $N$  next queries given a user’s current query  $q_t$ . The QFG method in (Boldi et al., 2008; Bonchi et al., 2012) learns a transition model of possible query reformulations within a session. The graph used for this baseline to predict next queries is constructed from the AOL query log (as in our proposed approach described in Section 7.2). From each user’s past query log history we construct a separate query flow graph. Given a user’s current query we first try to find the corresponding node in the graph. Then we sort the edges connected to that node, based on the weights. We choose the top  $N$



queries. We initially varied  $N$  from 2 to 8 in the interval of 3 and obtain the highest AP value when  $N$  was equal to 5. Hence we choose top 5 most likely queries from the graph. These queries are potential candidates for  $q_{t+1}$ . Then we use a LM-JM (Ponte and Croft, 1998) retrieval model with  $\lambda = 0.5$ , to retrieve a ranked list of the top 10 documents corresponding to each candidate next query. We then combine the 10 ranked lists into a single list of  $k$  predicted documents,  $L(\hat{q}_{t+1})$ , by the weighted CombSum method (Shaw et al., 1994). The weights used in the Combsum approach is proportional to the edge weights in the constructed query flow graph from AOL. Recall from Section 7.2, that the edge weights in the query flow graph estimates cooccurrence probability for a pair of queries within a search a session.

Note that this baseline relies on a user’s most recent query (i.e.  $q_t$ ) and only a user’s past queries (i.e.  $H_t$  comprising of only a user’s past queries). This method does not use similar queries of other users to proactively support the current user. Moreover this method also does not involve any embedding approach (e.g. graph representation learning) to determine the set of predicted next queries. Hence semantic similarity between a pair of words is not taken into account explicitly in this approach.

**Query Flow Graph ( $QFG^+$ ).** This method is the same as QFG, except that we construct a single query flow graph from all the past queries of all the users, rather than constructing separate query flow graphs for each user like QFG.

Note that this baseline relies on the current session context (i.e.  $C_t$ ) and the historical information from a query log (i.e.  $H$  comprising of both a user’s past queries and similar other user queries in past). This method also does not involve any embedding approach (e.g. graph representation learning) to determine the set of predicted next queries. Hence semantic similarity between a pair of words is not taken into account explicitly in this approach.

All the above mentioned baselines are text-only approaches which rely on learning task-query associations from exact term matches only, and are hence unable to address word-level semantic associations.

## Embedding Baselines

Now we discuss embedding based baseline approaches. The objective of using these baseline approaches is two fold. The first reason is to observe whether applying semantic similarity (in Equation 7.1) improves the proactive suggestion effectiveness. The second reason is to observe whether the embedding approach used in our method captures the semantic similarity better than existing embedding approaches for the effectiveness of proactive suggestions.

**Word-Embedding (WE).** The objective of using this baseline is to observe how an existing standard embedding approach, the skip-gram (Mikolov et al., 2013a) is effective for proactive suggestion. In this baseline, we use the **skip-gram** word embedding model (Mikolov et al., 2013a) on the training split of the AOL query log (i.e. first 20 million queries) to learn the semantic associations between terms in queries that are submitted close in time. We use the skip-gram embedding to find queries related to the current query  $q_t$ . Once we have a user’s current query, a user’s recent session history and related queries from the past, we use our proposed query formulation model (i.e. described in Equation 7.1) to retrieve the list  $L(\hat{q}_{t+1})$ . Since skip-gram word embedding does not directly learn representations of queries, we obtain the vector representation of a query by summing the vectors of its constituent words. The only difference between this approach with our proposed proactive suggestion model is that instead of using our graph based embedding, here we use the skip-gram embedding model.

**Hierarchical Recurrent Encoder Decoder (HRED).** The objective of using this as one of our baselines is to observe how an existing deep learning based query prediction model performs in proactive document suggestion. The HRED baseline method applies a sequence-to-sequence model for query prediction as proposed in (Sordoni et al., 2015). The encoder side is a hierarchical LSTM to model the sequential dependence between the terms within a query and also the queries within a session. Given the context vector obtained from the encoder of the hierarchical LSTM, the decoder part predicts the terms for the next query. Once we get the

predicted next query (i.e.  $q_{t+1}$ ), we retrieve documents with the predicted query using LM-JM similar to the previous baselines.

**Sequence-Sequence with Transformers (BERT).** BERT (Devlin et al., 2019) is a recent contextual embedding model which has proven to be effective in a variety of NLP tasks. The objective of using a BERT based model as one of our baselines is to compare the effectiveness of state-of-the-art contextual embedding approaches with our proposed embedding approach. The model makes extensive use of external information to construct a *masked* language model from large document collections with positional information. Specifically, we use the ‘BERT-Base model uncased’ model and fine-tune it on the AOL query log to predict the next query. Specifically, we used the ‘BERT-base-uncased’<sup>2</sup> pre-trained model as a knowledge source to learn the query generation sequence-sequence model using a standard LSTM architecture, similar in principle to (Futami et al., 2020; Rothe et al., 2020; Lewis et al., 2019; Nogueira and Cho, 2020). More precisely, in contrast to the hierarchical encoder-decoder architecture of HRED, for this baseline we feed in the dense vector (768 dimensional) for each query (in its entirety) obtained from a pre-trained BERT model (instead of feeding in the vectors for each of its constituent terms). Instead of HRED, for this method we use a single layer of LSTM cells to learn the sequence of queries directly. Similar to the HRED baseline, for this method during training, we feed in a sequence of each query (except the final one) within each session as inputs, the objective being to predict the final one as a reference output. The decoder part is similar to HRED comprising a softmax of dimensionality identical to the vocabulary size; the objective being to predict the likely sequence of words in a future query following a given partial session context.

**Paragraph Vector Embedding (D2V).** This baseline is similar to our proposed query formulation framework described in Section 7.2. The embedding similarity in Equation 7.1 is obtained from `doc2vec` (Le and Mikolov, 2014) in place of our proposed embedding approach. The ‘doc2vec’ method is first trained on AOL (i.e.

---

<sup>2</sup>[https://huggingface.co/transformers/pretrained\\_models.html](https://huggingface.co/transformers/pretrained_models.html)

on first 20 million queries of AOL log). Similar to our proposed graph-based embedding approach ‘doc2vec’ embeds both words and queries simultaneously. The reason for using this method as one of the baselines is to observe how our proposed query and word embedding approach performs compared to an existing word and query embedding approach.

**Non-negative Matrix Factorization (NMF).** In this baseline, we apply the non-negative matrix factorization (NMF) method (Salakhutdinov and Mnih, 2008) on the adjacency matrix of the weighted graph obtained with the method described in Section 7.2.1. With NMF we obtain a joint representation of both queries and words. Once we get the embedding for words and queries, we apply Equation 7.1 to formulate the proactive query. The reason for using NMF as one of our baselines is similar to D2V which is to observe how our proposed embedding method performs compared to state-of-the-art word and document embedding approaches.

For embedding approaches such as the WE and D2V approaches, the objective is to make each word similar to all the words appearing in its context. The size of context (i.e.  $W$ ) and the negative sample size (i.e.  $NS$ ) are common parameters in WE and D2V and our proposed embedding approach. We choose the optimal configuration as described in Section 5.5 in Chapter 5. The value for  $W$  is set to 5 and the value for  $NS$  is set to 10. For all the embedding based approaches, we initially varied the word vector dimension from 100 to 300 in the interval of 100 and observed the effectiveness of proactive suggestions in terms of AP in AOL test set described in Section 7.3.1. We obtained the best AP value for 200 dimensions. So we fixed the embedding dimension as 200 similar to (Grover and Leskovec, 2016; Sen et al., 2018b). the optimum values are reported in Table 7.3.

## Proposed Approaches

Unlike all the baseline approaches, our proposed proactive suggestion model uses a graph-based embedding approach which attempts to capture the task-based associations between words or queries. Hence our proposed approach is likely to be more

Method	$q_t$	$L(q_t)$	H	$C_t$	RL	T	WQ	JRL
RLM	✓	✓					✓	
QFG	✓		✓					✓
$QFG^+$	✓	✓	✓				✓	
SRLM	✓			✓		✓	✓	
CRLM	✓		✓	✓		✓	✓	
WE	✓		✓	✓	✓		✓	
HRED	✓		✓	✓	✓	✓	✓	
BERT	✓		✓	✓	✓		✓	✓
D2V	✓		✓	✓	✓	✓	✓	✓
NMF	✓		✓	✓	✓	✓	✓	✓
GRL	✓		✓	✓	✓	✓	✓	✓

Table 7.2: Summary of the approaches investigated. The columns denote sources of information, e.g. representation learning (RL), temporal information (T), weighted query (WQ), joint representation (JRL).

effective in formulating proactive queries that can eventually be used to provide proactive suggestions to the user. We investigate two variants of our proposed approach of graph representation based proactive IR. In the first variant, named GRLU (unweighted), we set the edge weights to 1, which means that our node embedding method only takes into account the reachability information from the connections between edges of a graph. In the second variant, named GRLW, we set the edge weights as described in Section 7.2.1.

Table 7.2 shows a summary of the sources of information used in each approach investigated in our experiments.

### 7.3.4 Parameter Setting

The training split of the AOL query log data is used to construct the graph (described in Section 7.3.1) which is used to obtain the embedding for both words and queries. The sessions in the test split of the AOL query log are used to evaluate the effectiveness of a proactive IR approach. For all the approaches, once we obtain  $\hat{q}_{t+1}$ , then we use LM-JM retrieval model to retrieve the top 10 documents as proactive suggestions (i.e.  $L_{\hat{q}_{t+1}}$ ) for the user. We use LM-JM with  $\lambda$  value 0.6. We use the same  $\lambda$  value for the retrieval using all the other approaches.

A common parameter in a test configuration of a proactive systems is the point at which the system becomes proactive. We refer to this parameter as  $\pi$ , where  $\pi$  is a number between 0 and 1. Since the number of queries varies across sessions, instead of using an integer value index we use a fraction value for  $\pi$ , which is interpreted as the proportion of queries in a test session for which an IR model behaves traditionally (i.e. for each user query it shows a single ranked list of documents rather than predicting a ranked list of documents which can be useful for the user in future). The smaller the value of  $\pi$ , the greater is the number of future queries for which anticipatory ranked lists are retrieved. For example, a value of  $\pi = 0.1$  for a session means that documents are retrieved proactively for 90% of the queries in this session.

To ensure fair comparisons between the approaches investigated, we optimize the parameter  $\pi$  of each individual method including baselines and proposed approaches independently.  $\pi$  is varied in the range  $[0.1, 0.9]$  to choose the optimal value of AP for each method. The optimal parameter settings for each method are also shown alongside each method in Table 7.3. The session length (i.e. the number of queries in a session) of any session is not known at first in the test set. So we use the average session length of all the search sessions in the training set multiplied by  $\pi$  to determine after how many queries in a search session in the test set a system will be proactive. For all the baseline and proposed approaches discussed in Section 7.3.3, we know that each one of the approaches eventually estimates a distribution of words given a user’s session context except QFG and  $QFG^+$ . We choose the top 5 terms to formulate the proactive query from the distribution. This proactive query is used to retrieve proactive document suggestions. However, recall from Section 7.3.2 that one of our evaluation metrics is QP-MRR, which measures the inverse of the rank of the user typed next query (i.e.  $q_{t+1}$ ) from a ranked list of predicted next queries (i.e.  $R(\hat{q}_{t+1})$ ). To formulate the  $i^{th}$  proactive query in  $R(\hat{q}_{t+1})$  of size  $k_1$ , we take the top 5 terms starting from  $i^{th}$  most important word obtained from the distribution estimated by the corresponding method. In our experimental setup, the value of  $k_1$  was set to 5, similar to (Sordoni et al., 2015). Note that the top most

Method		Optimal Parameter Settings						Metrics (Averaged over remaining $1 - \pi$ -th fraction of sessions)					
		$\pi$	$M$	$N$	$k$	$W$	$NS$	QP-MRR	$\rho$	MRR	AP	AP*	AP <sup>+</sup>
Text-based	RLM	0.7	10	-	-	-	-	0.2015	0.1510	0.1930	0.0931	0.1340	0.0628
	SRLM	0.7	10	-	-	-	-	0.2123	0.1615	0.2120	0.1035	0.1345	0.0895
	QFG	0.6	-	10	-	-	-	0.4233	0.1931	0.2415	0.1231	0.1671	0.0933
	$QFG^+$	0.6	10	10	-	-	-	<b>0.4510</b>	0.2311	0.2843	0.1561	0.1735	0.1231
	CRLM	0.6	-	-	30	-	-	0.4031	<b>0.2751</b>	<b>0.3821</b>	<b>0.1832</b>	<b>0.2881</b>	<b>0.1996</b>
Embedding & Seq2Seq based	WE	0.6	-	-	15	5	5	0.4899	0.2933	0.4550	0.2352	0.3183	0.2182
	HRED	0.6	-	-	15	-	-	0.5439	0.3011	0.4892	0.2471	0.3209	0.2153
	BERT	0.6	-	-	15	-	-	<b>0.5561</b>	<b>0.3103</b>	<b>0.4951</b>	<b>0.2486</b>	<b>0.3211</b>	0.2156
	D2V	0.6	-	-	15	5	5	0.3501	0.2244	0.3981	0.2123	0.2909	0.1991
	NMF	0.6	-	-	15	5	5	0.3501	0.2459	0.4081	0.2203	0.2785	<b>0.2254</b>
Our Proposed	GRLU	0.6	-	-	15	5	10	0.5532	0.2981	<u><b>0.5019</b></u> <sup>†</sup>	0.2471	0.3180	<u><b>0.2255</b></u>
	GRLW	0.6	-	-	15	5	10	<u><b>0.5764</b></u>	<u><b>0.3320</b></u> <sup>†</sup>	0.4775	<u><b>0.2518</b></u> <sup>†</sup>	<u><b>0.3285</b></u> <sup>†</sup>	0.2228

Table 7.3: Comparisons between the query prediction and document retrieval effectiveness (measured with QP-MRR and the accumulated document retrieval measures from column ‘ $\rho$ ’ to column AP<sup>+</sup>, respectively) obtained with different query prediction and proactive document retrieval methods. The best results for each group of methods (e.g. ‘text-based’) are bold-faced, and the best across all the groups are underlined. The ‘†’ symbol indicates statistical significance (paired t-test with 95% confidence) of our GRL variants in comparison to the best baseline approach, e.g. the value  $\rho = 0.3320$  (GRL-W) is significantly better than  $\rho = 0.3103$  (BERT).

predicted query is used to retrieve documents for proactive suggestion. The rest of the queries in  $R(\hat{q}_{t+1})$  are used only to measure the effectiveness of the proposed query prediction approaches.

## 7.4 Results and Discussion

In this section, we discuss the results reported in Table 7.3. We first examine the use of different information sources (i.e. a user’s recent session context, similar queries of the current user from that user’s past search log, similar queries of the other users from the past search log) to formulate proactive queries. We also examine the use of our proposed graph-based embedding approach to proactive document suggestion, compared to existing embedding approaches. Then we further analyze how different methods perform with different points of proactivity (i.e. different values of  $\pi$ ).

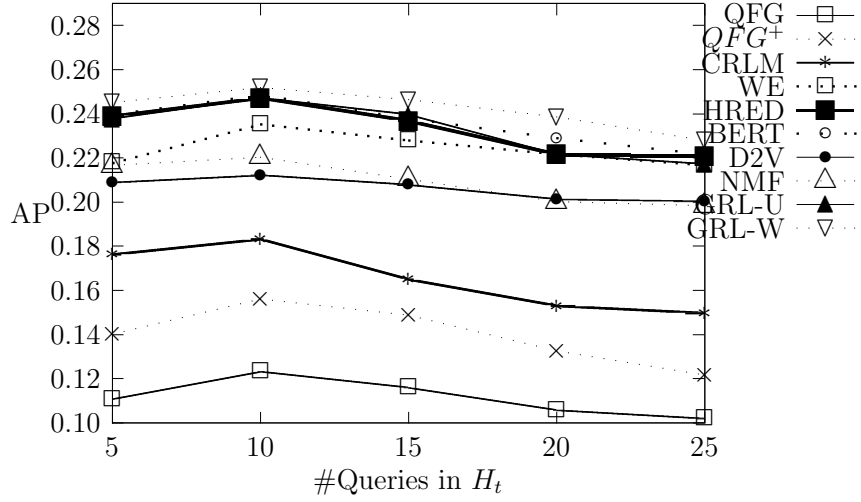


Figure 7.4: Sensitivity of proactive IR effectiveness with variations in the number queries considered to consider set  $H_t$ .

#### 7.4.1 Comparisons Between the Investigated Approaches

In Table 7.3, we report the results of applying different query formulation methods including baselines and proposed approaches on the 280,567 test sessions (i.e. mentioned in Table 7.1). All the different metrics reported are the average over all the 280,567 test sessions. For each session, the proactivity starts from the point  $\pi$  and then continues for each of the query entered by the user till the second last query in the session (i.e.  $q_{n-1}$ ). The observations from Table 7.3 are as follows.

##### Discussion on Text-Based Approaches

The top part of Table 7.3 shows the results for text-only approaches. Recall from Section 7.3.2 that QP-MRR measures the rank of the original  $q_{t+1}$ , in the predicted query list and the other evaluation metrics measure the similarity between the ranked list of documents for  $q_{t+1}$  and the predicted next query  $q'_{t+1}$ . The first interesting thing to observe from Table 7.3 is that for most of the text-based approaches an increase in QP-MRR value implies an increase in all other evaluation metrics (i.e.  $\rho$ , MRR, AP,  $AP^+$ ,  $AP^*$ ). So we can say that the more the predicted proactive queries are close to the original query, the more effective is the ranked list of document suggestion (i.e.  $L_{q'_{t+1}}$ ). The relatively poor results of RLM compared to SRLM and



CRLM for all of the evaluation metrics confirm the observation (reported in (Levine et al., 2017)) that the recent session context (i.e.  $C_t$  comprising of queries from  $q_1$  to  $q_t$ ) contains relevant information for predicting documents relevant to next likely information needs and that ignoring the context can yield poor proactive retrieval effectiveness. Similarly, CRLM performs better than SRLM, since CRLM uses all three information sources (i.e.  $C_t$ ,  $H_{u_t}$  and  $H_{ou_t}$ ), whereas SRLM uses only one information source (i.e.  $C_t$ ). QFG also produces better results than RLM because it uses the history from query logs. It can be seen that CRLM considerably improves the results in comparison to QFG and other text based approaches, which can be attributed to the fact that it uses information from both the context  $C_t$  and the history  $H_t$  (see Table 7.2), whereas  $QFG$  and  $QFG^+$  uses only  $H_t$ , RLM uses only  $q_t$  and CRLM uses only  $q_t$  and  $C_t$ . Hence we can say that using all the different information sources helps in both predicting proactive queries more accurate and also in proactive document suggestion. ' $QFG^+$ ' outperforms QFG and RLM for all metrics except MRR. This shows that using only a user's similar query from the past search log is less effective compared to using all the similar queries from different users in the past search log for proactive document suggestion. This confirms our hypothesis that using other users' similar activities helps in proactive suggestion.

### Discussion on Embedding Approaches

It can be seen that WE (skip-gram word vectors) mostly yields better results than D2V and NMF in terms of all the evaluation metrics. The reason for better results of WE compared to NMF can be explained by the fact that the deep learning based approach better captures the semantic similarity than a statistical approach in NMF. D2V embeds both words and queries simultaneously, whereas WE embeds only words. The query embedding in WE is obtained by summing the corresponding word vectors. The higher results for WE compared to D2V show that query vectors in WE capture the semantics better than D2V. BERT, due to its language model based pre-trained knowledge, turns out to be the best among the embedding baselines. NMF

performs slightly better than BERT only in terms of  $AP^+$  which measures how useful the proactive suggestion at time  $t$  is at providing new documents for the rest of the session (i.e. from  $q_{t+1}, \dots, q_n$ ). So it can be said that NMF is more useful in providing new relevant documents for suggestion compared to BERT. One possible reason for this is that the contextual embedding architecture means that the predictions using BERT are more biased towards creating queries which retrieve documents that the user has already seen in a search session.

### **Comparison Between Text-Based and Embedding based Approaches**

Generally speaking, embedding based methods (central and bottom-part of Table 7.3) yield better results than the text-only methods, most likely because these methods are able to make use of task-semantics driven associations between terms.

### **Discussion on Proposed Approaches**

It turns out that the weighted graph-based representation learning (i.e. GRLW) outperforms embedding approaches that embed either words alone (WE) or words with documents (D2V/NMF/BERT) in terms of all the evaluation metrics. This observation corroborates that a joint learning of the nodes of a graph (representing the session-driven co-occurrences between words and queries) turns out to be more effective than directly learning their representations. Apparently, the contexts obtained from neighbours in a graph in GRLW or GRLU are better able to model the possible information need transitions. The weighted version of GRL (GRLW) mostly outperforms the unweighted scenario, which means that the edge weights yield better representations of the nodes in an embedded space. The unweighted case, GRLU, is however better able to retrieve novel documents, as measured by the aggregated rewards with  $AP^+$ . One possible reason for this is that GRLU assigns equal probability to every query occurring within a single session to estimate the next query, so the chances of it appearing in diverse queries in the next query prediction for GRLU is greater than would be the case for GRLW where weights are

not uniform.

We performed a paired t-test to compare between the two variants of GRL (i.e. GRL-U and GRL-W) and the best baseline approach reported in Table 7.3 (i.e. BERT). The results for all the metrics were significantly better with with 95% confidence interval.

## 7.4.2 Further Analysis

Here we describe the effectiveness of proactive suggestion with respect to different parameters. First we focus on the impact on  $AP$  value of variation in the proactive index (i.e.  $\pi$ ) with all the approaches described in Table 7.3.

### Effectiveness of Proactive Suggestions with the variation of Proactive Index (i.e. $\pi$ )

Since  $\pi$  is a common parameter determining how early in a search session a proactive search system becomes proactive, we further analyze if GRL consistently out-

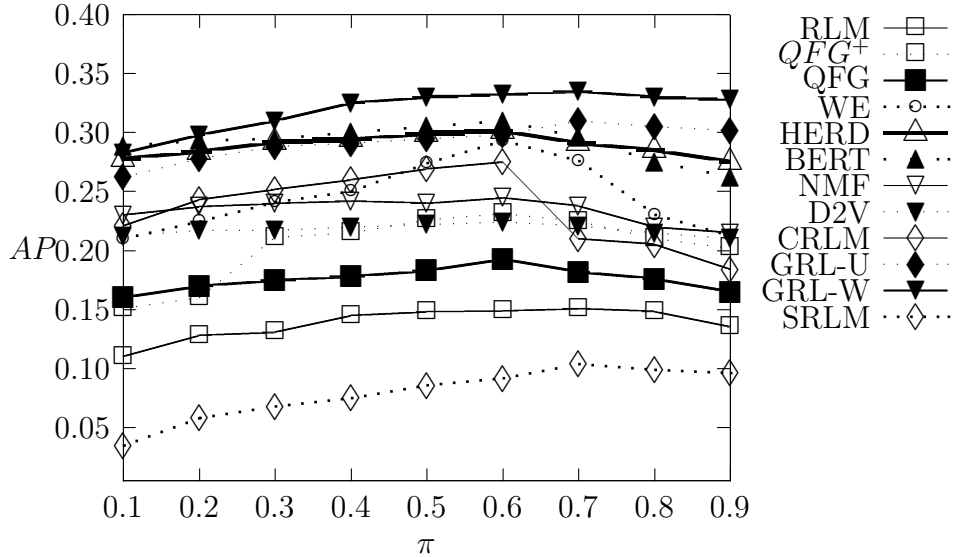


Figure 7.5: Sensitivity of proactive IR effectiveness with variations in  $\pi$ , i.e., relative proactivity duration.

performs the baselines over a range of different configurations for  $\pi$ . In Figure 7.5 we show the variations in AP values across different methods with the values of  $\pi$

ranging from 0.1 to 1. From Figure 7.5, we observe that GRLW is able to consistently predict the relevant documents for subsequent information needs, except in a small number of cases where either WE or GRLU perform marginally better.

### Effectiveness of a User's Own Search History vs. Other Users' Search History in Proactive Suggestions

We further analyse how much importance should be given a user's own past queries compared to other users' similar queries. We introduce a hyperparameter  $\beta$  that controls the importance of a user's own past queries compared to other users' similar queries from past query log. We modify the query formulation Equation 7.1 to introduce  $\beta$  into it. While estimating  $p(w/\theta_R)$ , if  $w \in H_{u_t}$  then we multiply  $\beta$  with it, and if  $w \in H_{ou_t}$ , we multiply it with  $1 - \beta$ . This is shown mathematically in Equation 7.4.2.

$$P(w/\theta_R, a_t) = \begin{cases} \prod_{t \in a_t} \sum_{a \in B_t} P_e(t|a) P_e(w|a) e^{-\delta(a_t, a)^2 \frac{\mathbf{a}_t \cdot \mathbf{a}}{|\mathbf{a}| |\mathbf{a}_t|}} \beta & \text{if } w \in H_{u_t} \\ \prod_{t \in a_t} \sum_{a \in B_t} P_e(t|a) P_e(w|a) e^{-\delta(a_t, a)^2 \frac{\mathbf{a}_t \cdot \mathbf{a}}{|\mathbf{a}| |\mathbf{a}_t|}} (1 - \beta) & \text{if } w \in H_{ou_t} \end{cases}$$

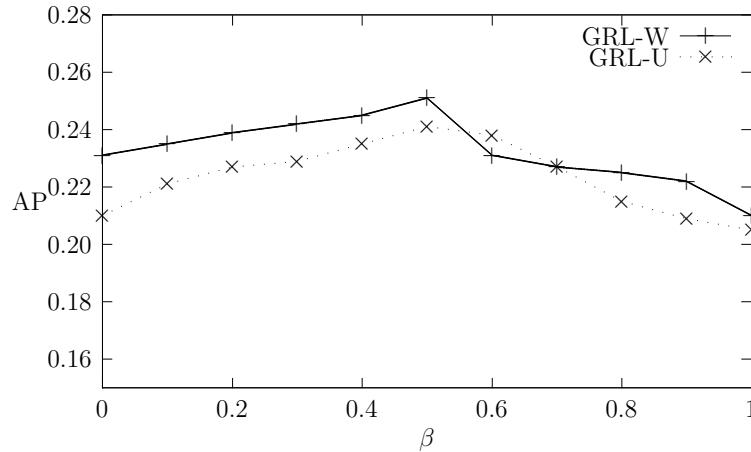


Figure 7.6: Sensitivity of proactive IR effectiveness with variations in  $\beta$ , i.e., importance of user's own past queries compared to similar queries from other users.

The higher the value of  $\beta$ , the higher is the importance of a user's own query compared to the other users' query and vice versa. We varied the value of  $\beta \in [0, 1]$

with the intervals of 0.1 for our proposed approach. Figure 7.6 shows the change in the values of evaluation metric AP with the variation of  $\beta$ . We found the best results for  $\beta = 0.5$  (i.e. when equal importance is given to both the sources of similar queries). One likely reason for this result is while computing the embedding we focus on capturing task based semantic associations based on the co-occurrences of words and queries. So while computing similar queries it does not matter whether it comes from the user or other users.

### Effect of Number of Query Terms in Proactive Suggestion Effectiveness

In Figure 7.7 we show the change in proactive suggestion effectiveness in terms of AP (described in Section 7.3.2) with the number of query terms used for proactive query formulation. For each of the methods, maximum value of AP is observed when the number of query terms were 5. All the metrics of all the methods reported in Table 7.3 are also computed on a proactive query formulated using 5 query terms. Similarly, in Figure 7.4 we show the proactive suggestion effectiveness in terms of

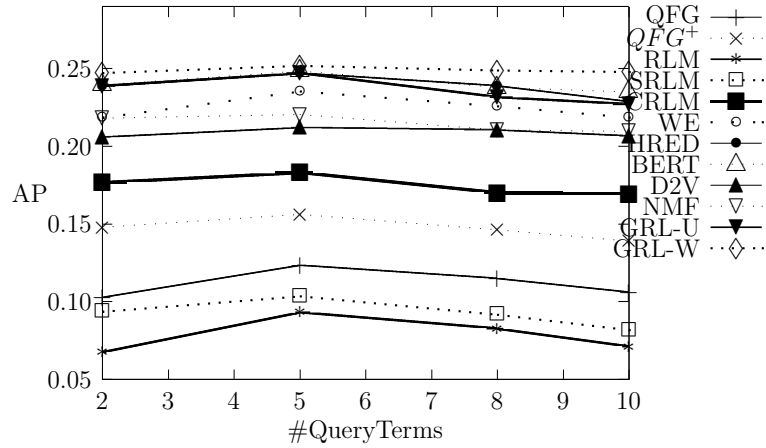


Figure 7.7: Sensitivity of proactive IR effectiveness with variations in the number of terms in proactive query.

how much history data (i.e.  $H_t$ ) we should consider for proactive suggestion. The best results are obtained when the number of queries in  $H_t$  were 10. All the metrics of all the methods reported in Table 7.3 are also computed using 10 queries from the past.

Query prediction is likely to be easier for the scenario where the next query is similar to the current query (i.e. there is significant word overlap between the current query and the next query). A proactive suggestion method should work well in all the cases (i.e. when similarity of the next query with the previous one is either high or low). We investigate the issue below.

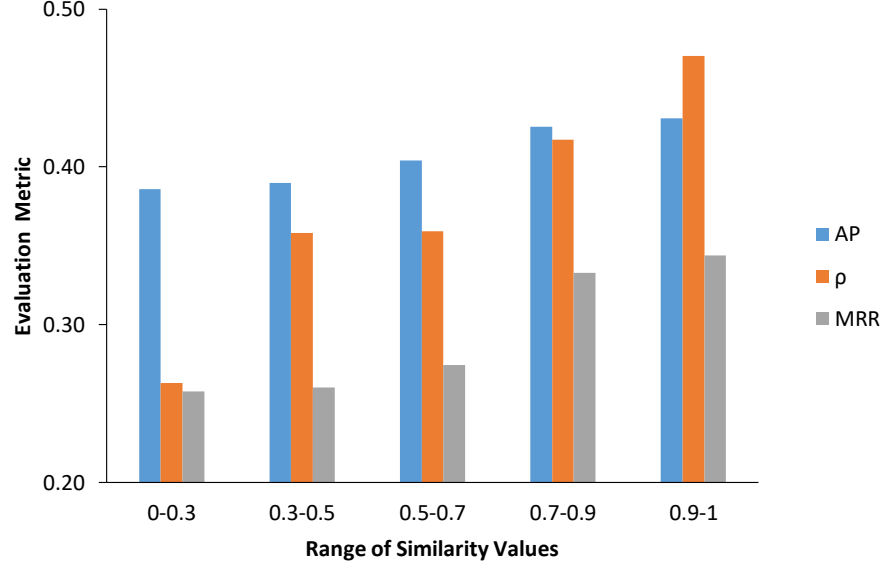


Figure 7.8: AP,  $\rho$  and MRR values of GRLW for different ranges of similarities between consecutive queries.

### Effectiveness of Proactive Suggestions With the Similarity of Previous Queries

We wanted to examine how effectively our proactive suggestion works when the next query is not very similar to a user’s current query. To test the effectiveness of the best performing proactive IR model in our experiments (GRLW) for all cases, we segmented the test set queries into different ranges of cosine similarities between consecutive queries, and present the AP,  $\rho$  and MRR values for the different similarity ranges in Figure 7.8. For example, the left-most range of Figure 7.8 represents all instances when the cosine similarity between the (true) next query,  $q_{t+1}$ , and the current query,  $q_t$ , is between  $[0, 0.3)$  and is thus representative of considerable shifts in information need between the queries. These instances are thus likely to

Method	Metrics				
	$\rho$	MRR	AP	AP*	AP <sup>+</sup>
GRL-U	0.2981	0.5019	0.2471	0.3180	0.2255
GRL-U + BERT	0.3012	<b>0.5021<sup>†</sup></b>	0.2502	0.3195	<b>0.2261</b>
GRL-W	0.3320	0.4775	0.2518	0.3285	0.2228
GRL-W + BERT	<b>0.3351<sup>†</sup></b>	0.4791	<b>0.2601<sup>†</sup></b>	<b>0.3291</b>	0.2231

Table 7.4: Investigating BERT-based passage reranking on retrieval effectiveness obtained with our proposed approaches (the best performing ones in Table 7.3). Similar to Table 7.3, the metric values are averaged over remaining  $1 - \pi$ -th fraction of sessions. The notation <sup>†</sup> indicates significant improvements (paired  $t$ -test with 95% confidence) obtained with the reranking step.

be the most difficult cases for a proactive IR model. It can be seen from Figure 7.8 that GRLW works fairly well even for these cases involving significant changes of information need. As expected, the performance of the proactive IR model is better (i.e. 0.9-1 and 0.7-0.9 ranges in Figure 7.8) if the information needs are more directly related, i.e. with overlap between query terms indicating potential query specifications or generalizations (Carterette et al., 2014).

### Effect of BERT-based Re-ranking on Proactive Suggestion Effectiveness

In this section, following the exposition of some of the recent work which has reported enhanced retrieval effectiveness with the application of BERT-based similarity computation between embedded vectors of documents (passages) and queries (Yilmaz et al., 2019; Khattab and Zaharia, 2020), we investigate if such a reranking step can further improve the effectiveness of proactive document retrieval.

The methodology for BERT based reranking is described as follows. After retrieving the top 10 documents with our proposed method GRL (CRLM with joint query-word embedding), we partitioned each document into chunks of passages of length 512 characters with word wrapping across the chunks. We then computed the aggregate (specifically, maximum) of the cosine similarities of pre-trained BERT vectors (768 dimensional) for these passages with respect to the query BERT vector (also 768 dimensional). These aggregated passage-query similarity scores for each

document were then used to rerank the documents. The methodology that we employ is similar in principle to ColBERT (Khattab and Zaharia, 2020). The only difference of our approach with ColBERT is that we did not conduct an end-end training with triples of the form  $(q, d_r, d_n)$  -  $q$  a query,  $d_r$  and  $d_n$  a relevant and a non-relevant document, respectively, because of the lack of availability of relevance assessments or clicked document information (recall that we did not use the information from clicked URLs in our experiments).

Even without the end-end training, we observe from the results presented in Table 7.4 that BERT vectors were useful to further improve the retrieval effectiveness in terms of most of the metrics.

## 7.5 Concluding Remarks

In this chapter we addressed our fourth research question RQ4 ‘Can we provide proactive suggestion by leveraging similar activities of other users’. We used the generative proactive formulation framework proposed in Chapter 3 to retrieve documents proactively during a search session. The results conclude that the collaborative information used from similar activities of other users can be used in generating effective proactive document suggestions during search sessions.



# Chapter 8

## Conclusions

In this thesis we examined proactive IR in two different settings: a Single Stage task scenario and a Multi Stage task Scenario. Single stage scenario refers to the tasks where a user is interested about a single topic. Multi-stage scenario refers to all those tasks which has many sub-tasks in it. In this chapter, we summarize the overall and individual contributions of this study and outline potential directions for future work.

### 8.1 Research Questions Revisited

In this section, we revisit the research questions, introduced in Chapter 1, and summarize how each one of them has been addressed in the following chapters.

#### 8.1.1 Proactive IR Framework

There is no standard framework of proactive IR in research literature. In Chapter 3 we introduced a generic framework for proactive IR which is implemented in Chapter 6 and Chapter 7 for single-stage task and multi-stage task scenario respectively. In our proposed framework, we considered user activities that have word based representation. Another assumption of our framework is that a proactive IR model will show proactive suggestions to a user after a fixed number of interval of activities.

Once the framework is established we proposed a proactive query formulation model (i.e. addressing the third research question) integrating the different information sources available to the proactive IR model.

### **8.1.2 Evaluating Proactive IR**

In Chapter 4, we addressed RQ2 which is ‘how we can evaluate a proactive IR model’. We proposed an evaluation framework based on the proactive IR setup, described in Chapter 3, to measure the effectiveness of proactive IR models. This evaluation framework has been used in Chapter 7 and 6 to measure the effectiveness of proactive suggestions in single stage and multi stage scenario.

### **8.1.3 Grouping Similar activities of Users**

We address our second research question ‘How can we group user activities related to similar information goal?’ in Chapter 5. We proposed a graph-based embedding approach that takes into account the temporal and tempo-lexical contexts of activities to learn task-specific semantics between a pair of words or activities. Our experiments on the AOL query log indicate that the proposed temporal and tempo-lexical embedding method outperforms a baseline word2vec embedding approach and other techniques. Once we can identify activities similar to a user’s current activity using our proposed embedding approach, we can use these similar activities to formulate proactive queries using the proactive query formulation method framework described in Chapter 3. We examine the use of embedding approaches for proactive query formulation in Chapter 7.

### **8.1.4 Proactive Support for Single Stage Tasks**

In Chapter 6 we showed how we can proactively support a user in a single stage task scenario with our proposed proactive query formulation model described in Chapter 3. We first described a simulation setup based on the TREC Novelty track

dataset. In the simulation setup, we show how a user interested in one of the TREC Novelty track topics could interact in a desktop environment. Our objective was to proactively suggest to the simulated user a ranked list of documents at different stages of user interactions with the desktop environment while carrying out their task. We use our proposed proactive query formulation model to retrieve a ranked list of suggestions for the user. We reported an ablation study which analysed the importance of different components in our proposed query formulation model. We also showed the variation in the effectiveness of the proactive suggestions with different levels of initial knowledge of the simulated users.

### **8.1.5 Proactive support in Multi-stage Task Scenario**

In Chapter 7 we showed how we can proactively support a user in multi-stage task scenario. More specifically, we addressed our fourth research question ‘How we can proactively support a user using similar activities of other users?’ For experimental purposes, we considered only search activities (i.e. search queries) of the users. We again used AOL search query log in our experiments. Given the current query of a user within a session our objective was to predict a ranked list of documents that could help the user in performing his overall task in the rest of a session. We used our embedding model proposed in Chapter 5, to bring similar queries of other users to formulate a proactive query which is used to retrieve a proactive list of suggestions for the user. From the results of the experiments, we can conclude that using similar queries of other users makes a proactive IR model more effective rather than considering only a user’s own history and current search context.

## **8.2 Future Work**

In this thesis we explored how we can use relevance model estimation along with embedding techniques to proactively support a user in their task. There are a number of possible directions which can be further explored. We introduce each one

of these in this section.

### **8.2.1 Chapter 3**

In this chapter we proposed a proactive suggestion framework for proactive IR. One of the assumptions in our proposed framework is that a proactive IR model will provide proactive suggestions after a fixed number of user activities. Future work in this direction should involve investigation of identification of opportunities for a proactive IR model to provide suggestions to the user in a more reliable context dependent manner.

For our proactive suggestion framework, we focused on how we can provide proactive suggestions to the user in terms of a ranked list of documents. A possible future extension of this could involve investigating the format of proactive suggestions e.g. document title, document snippet.

### **8.2.2 Chapter 4**

In this chapter we proposed an evaluation framework for proactive IR. We focused on producing evaluation metric that is more focused on measuring the quality of the proactive query or the retrieved documents. Since the objective of proactive IR is to provide information sources to the user that is eventually useful for him to complete his current task. One future direction is to incorporate user experience in evaluation metric. The two different possible attributes which can add value to an existing evaluation metric are described below.

- Usefulness of the proactive suggestions to the user.
- User satisfaction measured over all the proactive suggestions for a task.

### **8.2.3 Chapter 5**

In this chapter we proposed a graph-based embedding technique which can be used to group user activities related to the same information goal together. Possible future

work in this direction could be to explore embedding techniques which can find different information needs associated with a single information goal. For example, if we have activities related to vacation planning, then the embedding technique would be able to group different aspects (e.g. booking tickets, planning accommodation, places to explore) of vacation planning separately.

Moreover, another future work direction could be how we might use external knowledge to enhance existing embeddings to group user activities.

### **8.2.4 Chapter 6**

This chapter discussed creation of proactive suggestions in a simulated personalized desktop setup. Future work could be related to user studies to evaluate the performance of proactive suggestions in a more realistic setting.

Moreover, another interesting future work direction could involve finding ways to incorporate external knowledge bases to improve proactive suggestions to the user.

### **8.2.5 Chapter 7**

This chapter described use of proactive suggestion models in a web search scenario. In our proposed proactive IR setup, for each query typed by a user, two ranked lists were provided. One is for a user's current query and the other one is for the next information need that can be in user's mind. One possible future work in this direction could be to carry out real user studies to investigate what should be ideal format (e.g. both ranked lists combined or separate ranked lists) to show proactive suggestions to the user.

Moreover, another future direction can be use of click information of the real users as a part of the creation of the proactive suggestion lists to estimate a user's next information need in a better way.

## 8.3 Closing Remarks

We believe that the work presented in this thesis has opened potential new research directions for proactive information retrieval not only in developing proactive query formulation methods, but also in comparing proactive IR models in a reproducible setup. We hope that this work will act as a starting point for other researchers to continue investigations on the problems that we addressed in an endeavour to further improve the techniques presented in this thesis and find further applications for them.

# Appendices

# Appendix A

## Publications

The research presented in this dissertation was published in several peer-reviewed conference proceedings. The work presented in Chapter 3 has been published in (Sen et al., 2018a). The work presented in Chapter 5 was published in (Sen et al., 2018b) and (Sen et al., 2019).

### A.1 My Publications

1. P.Sen, D. Ganguly and G.J.F. Jones, Procrastination is the Thief of Time: Evaluating the Effectiveness of Proactive Search Systems., In Proceedings of SIGIR 2018, Ann Arbor, USA, July 2018, pp: 1157-1160.
2. P.Sen, D. Ganguly and G.J.F. Jones, Tempo-Lexical Context Driven Word Embedding for Cross-Session Search Task Extraction, In Proceedings of NAACL 2018, New Orleans, USA, July 2018, pp: 283-292.
3. P.Sen, D. Ganguly and G.J.F. Jones, Word-Node2Vec: Improving Word Embedding with Document-Level Non-Local Word Co-occurrences, In Proceedings of NAACL 2019, Minneapolis, Minnesota, USA, July 2018, pp: 1041-1051.
4. P.Sen, D. Ganguly and G.J.F. Jones, I know what you need: Investigating Document Retrieval Effectiveness with Partial Session Contexts. (under re-



view in ACM Transactions on Information Systems)

also have a number of publications that are explicitly not linked to the topic of my PhD. These publications are listed below.

1. P.Sen, D. Ganguly Towards Socially Responsible AI: Cognitive Bias-Aware Multi-Objective Learning, In Proceedings of AAAI 2020, New York, USA, February 2020, pp: 2685–2692.
2. P.Sen, D. Ganguly and G.J.F. Jones, The Curious Case of IR Explainability: Explaining Document Scores within and across Ranking Models, In Proceedings of SIGIR 2020, Virtual Event, China, July 2020, pp: 2069-2072.
3. Gabriella Pasi, Gareth J. F. Jones, Keith Curtis, Stefania Marrara, Camilla Sanvitto, Debasis Ganguly, **Procheta Sen** Overview of the CLEF 2018 Personalised Information Retrieval Lab (PIR-CLEF 2018)
4. Gabriella Pasi, Gareth J. F. Jones, Stefania Marrara, Camilla Sanvitto, Debasis Ganguly, **Procheta Sen** Evaluation of Personalised Information Retrieval at CLEF 2017 (PIR-CLEF): Towards a Reproducible Evaluation Framework for PIR

# Appendix B

## Further Analysis of Proactive IR Evaluation Metric

### B.1 Introduction

We first describe three sample proactive IR systems with an intuitive preferred ranking order. The objective of the experiments is to compute the values of our proposed metric (Equation 4.7 in Chapter 4) on these systems, and see if the ranking induced by our metric corresponds to the intuitive preference among the systems. We denote the proposed metric as PREVAL in the rest of this chapter.

#### B.1.1 Sample Proactive Search Systems (PSS)

While developing a PSS is not the objective here, for the sake of comparisons between different systems, we describe two simple PSS approaches. We subsequently compute our proposed PSS effectiveness metrics under a number of different settings.

#### B.1.2 Relevance Model based PSS (RM-PSS)

The first approach employs a relevance model Lavrenko and Croft (2001) to obtain the predicted ranked list  $L(q'_i)$  given the previous query  $q_{i-1}$  and the ranked list retrieved in response to it, namely  $L(q_{i-1})$ . The relevance model employs relevance

feedback to improve retrieval effectiveness. In this case, we argue that the expanded query obtained from the estimated distribution  $P(w|Q)$  (see Lavrenko and Croft (2001) for details on the notations) is representative of a reformulated query, which is then used to retrieve the recommended documents  $L(q'_i)$ . We call this system RM-PSS (Relevance Model based PSS).

### B.1.3 Query Prediction based PSS (QP-PSS)

The second PSS approach employs a supervised query prediction algorithm to compute a set of most likely query reformulations given a current query. The probability of query reformulations is trained by constructing a query flow graph (QFG) from a large number of real-user queries in a query log Boldi et al. (2008). In a QFG, each node represents a query and the weight of each edge  $(q_i, q_j)$  indicates the likelihood of  $q_j$  following  $q_i$  within a search session. This QP-PSS based system obtains a ranked list of  $k$  predicted next queries,  $R_k(q_{i-1})$ , from the current query  $q_{i-1}$ , sorted in descending order by the transition probabilities estimated from a QFG. It then retrieves documents with each predicted query  $q \in R_k(q_{i-1})$  and outputs a weighted COMBSUM Shaw et al. (1994) of the lists. We experiment with two variants of COMBSUM weights, one with uniform weights, denoted as **QP-PSS-U**, and the other with weights set to probabilities of reformulation estimated from the QFG, denoted as **QP-PSS-W**. With respect to the three approaches described, the intuitive ranking between them should be **RM-PSS** < **QP-PSS-U** < **QP-PSS-W**.

### B.1.4 Experiment Settings and Results

We used a subset of the AOL query log data from the period of Mar'06 to Apr'06 for our experiments, which comprises over 21M queries. The first 20M queries (preprocessed using a Porter stemmer) were used as a training set to construct a QFG, which was then used in the QP-PSS approaches. A session length of 26 minutes, as prescribed in Lucchese et al. (2013), was used as the adjacency criteria

Table B.1: PREVAL metrics computed over the evaluation set.

Proactive System	PREVAL-RR	PREVAL- $\rho$
RM-PSS	0.0235	0.4955
QP-PSS-U	0.0503	0.4980
QP-PSS-W	0.0503	0.5004

to construct the CFG. As our evaluation set to measure the relative performances of the three PSS systems, we use a random subsample of 50 query sessions. Each session of the evaluation set comprises at least 5 queries (after removing duplicate queries within a session). The average session length in our evaluation set is 8.12.

Table B.1 shows the results of applying the two variants of the metric PREVAL on the test set of 50 query sessions. To compute the reward functions,  $M$  was set to 10 (see Equation 4.6). It can be seen that RM-PSS performs the worst in terms of both the RR and  $\rho$ -based PREVAL scores. This is to be expected since RM-PSS only uses the information from the current query and the top documents retrieved in response to it. Our metric is also able to rank QP-PSS-W better than QP-PSS-U. This is also expected since the latter considers the contribution from all predicted queries uniformly when constituting the final list of suggested documents. An interesting observation is that PREVAL- $\rho$  is better able to distinguish between QP-PSS-W and QP-PSS-U (in terms of the relative differences between the scores) in comparison to PREVAL-RR. This shows that the rank correlation based reward acts as a better estimate for PSS effectiveness.

Figure B.1 shows the relative differences between the PREVAL-RR values for systems A (RM-PSS) and B (QP-PSS-W), i.e.,  $\delta = \frac{P_A - P_B}{\max(P_A, P_B)}$ , for a range of  $\pi$ -index values  $\pi_A$  and  $\pi_B$ . To report an average over sessions of varying lengths, we plot relative proportions of  $\pi_A$  and  $\pi_B$  values within a range of 0.1 to 0.9. A sample cell  $(\pi_A, \pi_B) = (0.2, 0.6)$  in both plots of Figure B.1 indicates that A started suggesting documents after 20% queries within a session were executed, whereas B waited until 60% of the queries had been executed before initiating proactive suggestions. Each of the  $9 \times 9 = 81$  cells of Figure B.1 represents an experimental observation. The left

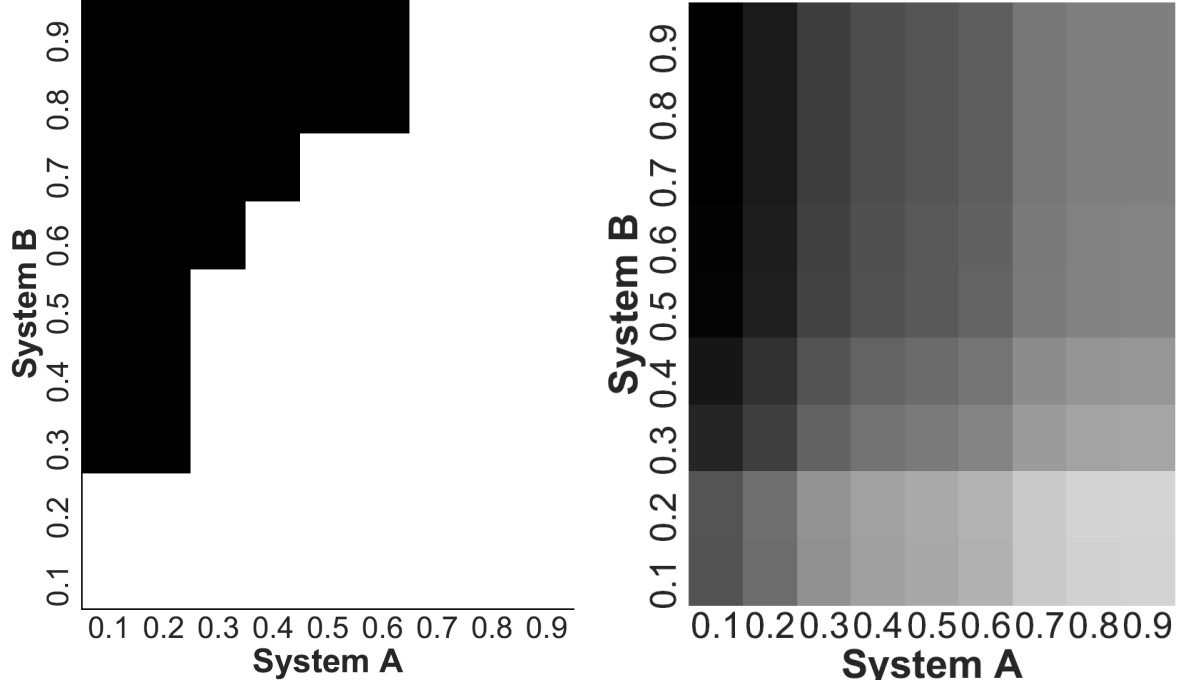


Figure B.1: Comparison of PREVAL-RR values of RM-PSS (x-axis) and QP-PSS-W (y-axis),  $\pi$ -index values being shown alongside the axes. i) Left: A black cell indicates that  $P_A > P_B$ . ii) Right: Each cell plots the value of  $\frac{P_A - P_B}{\max(P_A, P_B)} \in [-1, 1]$ , darker shades denoting values towards 1 ( $P_A > P_B$ ).

plot indicates whether RM-PSS is the winner (black cell). The right plot shows the relative differences between the PREVAL-RR values of RM-PSS and QP-PSS-W. If RM-PSS performs better, this relative difference is close to 1 (represented by darker shades), whereas lighter shades indicate that QP-PSS-W is better.

It can be seen that if QP-PSS-W starts predicting too late in comparison to RM-PSS (cells corresponding to the upper left region of the plots in Figure B.1), RM-PSS is the winner. This happens because the rewards accumulated from a small number of proactive suggestions in these cases is not sufficient to outweigh the effectiveness of the early suggestions coming from RM-PSS. This in turn shows that the metric is able to prefer cold-start proactive systems. On the other hand, the system QP-PSS-W (being a more effective system) often wins in the central and the bottom-right part of the plots in Figure B.1. This can be seen from the white regions in the left plot, and the lighter shades in the right plot of Figure B.1, with the relative differences getting larger towards the bottom-right. This shows that the

metric is not overly biased towards cold-start systems and can favour more effective systems with higher  $\pi$ -index values.

# Appendix C

## Evaluation of Proposed Word Embedding in Standard Task

### C.1 Experimental Setup

In this section, we describe our experimental setup to evaluate our proposed word embedding method with state-of-the-art word embedding approaches.

#### C.1.1 Dataset

A word embedding algorithm requires a collection to learn word representations. To compare the various word embedding approaches (i.e. our method and the baselines), we use the DBPedia (2014) corpus, which is a collection of abstracts of Wikipedia pages crawled in 2014<sup>1</sup>. Dataset characteristics are outlined in Table C.1. As part of pre-processing, we removed words with collection frequency less than 10 and also removed stopwords<sup>2</sup>.

---

<sup>1</sup>[http://downloads.dbpedia.org/2014/en/long\\_abstracts\\_en.ttl.bz2](http://downloads.dbpedia.org/2014/en/long_abstracts_en.ttl.bz2)

<sup>2</sup><http://www.lextek.com/manuals/onix/stopwords2.html>

#Documents	4,641,754
#Avg. Doc Length (#words)	43.23
#Vocabulary size	461,572
#Tokens	202,575,916

Table C.1: Dataset characteristics of DBPedia-2014.

### C.1.2 Baselines

We use three state-of-the-art embedding approaches namely skip-gram word2vec with negative sampling (SGNS) (Mikolov et al., 2013a), Glove (Pennington et al., 2014) and Fasttext (Joulin et al., 2016). All these methods rely only on co-occurrences (at the level of words for the first two and at the level of character n-grams for the last one) within a word or character n-gram window of specified length  $k$  (acting as a parameter). Fasttext learns the vector representation of each word by aggregating (vector sum) the vector representations of its constituent n-grams.

Additionally, we also employ a more recent approach, namely ELMO (Peters et al., 2018), which relies on a pre-trained model (comprised of stacked bidirectional LSTMs) to infer vectors for a given context (typically a sequence of words). For our experiments,

Although not an embedding approach, the LDA topic modeling algorithm outputs two matrices, namely  $\theta \in \mathbb{R}^{M \times d}$  and  $\phi \in \mathbb{R}^{d \times V}$ , representing the document-topic and topic-words distribution respectively (Blei et al., 2003). LDA uses document-level word co-occurrences to estimate both these matrices. In principle, one can then use the  $\phi$  matrix as a substitute for the word embedding parameter matrix of SGNS (see Equation 5.1). This gives  $d$  dimensional vectors for each word purely with a global co-occurrence based approach.

## C.2 Implementation Details

The number of nodes in the graph constructed for our proposed embedding approach is equal to the number of unique words in the collection of DBPedia. We removed



any word that has appeared less than 5 times in the collection. We connect a pair of words if they have co-occurred in a DBpedia article. Next we described how exactly we computed weights between a pair of words.

**Co-occurrence Weights.** The weight  $\omega(x_w, x_u)$  between word-nodes  $x_w$  and  $x_u$  is intended to represent a measure of association between these words. The first step is to include non-local co-occurrences to accommodate weighted document-level co-occurrences between a pair of words. Instead of considering the collection  $C = \{t_1, \dots, t_T\}$  as a stream of words, we consider  $C$  as a set of  $M$  documents  $\{D_i\}_{i=1}^M$ .

We compute the co-occurrence probability of two words  $w$  and  $u$  as

$$P(w, u) = \frac{\sum_{i=1}^M \mathbb{I}(w, u, D_i)}{\sum_{i=1}^M \mathbb{I}(w, D_i) \sum_{i=1}^M \mathbb{I}(u, D_i)}, \quad (\text{C.1})$$

where the numerator denotes the total number of times that the words  $w$  and  $u$  co-occur in the collection of all documents, and the denominator denotes the number of times each occur independently.

In our approach, we use a generalized form of Equation C.1, where analogous to the Jelinek-Mercer smoothing method Ponte and Croft (1998), we take into account the informativeness of the co-occurrences by linearly combining the frequencies with the global statistics of inverse collection frequency. More specifically,

$$P_\alpha(w, u) = \alpha P(w, u) + \frac{(1 - \alpha)T^2}{|\Lambda(w)||\Lambda(u)|}, \quad (\text{C.2})$$

where  $P(w, u)$  represents the maximum likelihood estimate computed by Equation C.1 and the denominator denotes the product of the collection frequencies of the terms. It can be seen that Equation C.2 allows relative weighting of the term frequency and the informativeness components.

**Combination with Local Co-occurrences.** The next step in our word-node2vec method is to augment the non-local co-occurrence information computed as per

Equation C.2 with the local co-occurrence of SGNS as defined in Equation 5.6. For this, analogous to Pennington et al. (2014), we compute the probability of co-occurrence between a word pair restricted within a window of size  $k$  over the whole collection. More formally,

$$P_k(w, u) = \frac{1}{|\Lambda(w)|} \sum_{i \in \Lambda(w)} \mathbb{I}(t_{i+j} = u)_{j=-k}^k \quad (\text{C.3})$$

Next, we assign weight to an edge by combining the local and non-local co-occurrence probabilities estimated from Equations C.3 and C.2 respectively. Formally speaking,

$$\omega(x_w, x_u) = P_\alpha(w, u)P_k(w, u). \quad (\text{C.4})$$

## C.3 Evaluation Tasks and Datasets

We denote our proposed word embedding approach as word-node2vec. To compare the relative performance of our proposed word embedding approach with the existing word embedding baselines, we use a number of datasets, each corresponding to one of the following three evaluation tasks.

### C.3.1 Word Similarity.

A standard way to measure the effectiveness of embedded words is to measure how well the similarity between a pair of words correlates with human judgments. Two such standard datasets that we use for our experiments are the WSIM-353 (Finkelstein et al., 2014) and the MEN (Bruni et al., 2014) datasets. Both comprise a list of word pairs, with an associated human judged similarity value. This similarity value is expected to be high for semantically similar words, such as ‘morning’ and ‘sunrise’ (human assigned score of 49 out of 50), and low for semantically unrelated words, such as ‘angel’ and ‘gasoline’ (score of 1 out of 50), both examples being taken from the MEN dataset.

Dataset	Composition	Example
MSR	Syntactic	good:better rough:X
Google	Syntactic and Semantic	Athens:Greece Berlin:X
SemEval	Syntactic and Semantic	dog:bone bird:X

Table C.2: Word analogy datasets overview.

### C.3.2 Word Analogy.

The word analogy task consists of templates of the form “A:B as C:X”, where A, B, and C are given words, whereas X is unknown. Using a vector representation of words this analogy task is solved by retrieving the vector most similar to that of  $\mathbf{B} + \mathbf{C} - \mathbf{A}$ . A word embedding is considered effective if it finds a greater number of correct answers (resulting in higher accuracy).

We employed three different analogy datasets, namely, the Google Analogy (Mikolov et al., 2013a), the MSR Analogy (Mikolov et al., 2013b) and the SemEval-2012 task 2 (Jurgens et al., 2012) datasets. The MSR dataset contains syntactic questions only involving morphological variations. The Google dataset on the other hand contains both syntactic and semantic questions.

Given an analogy ‘A:B as C:D’, the Semeval-2012 task requires prediction of the degree to which the semantic relations between A and B are similar to those between C and D. In our experiments, we treat the given entity D as unknown and seek to predict D, similar to the MSR and Google analogy datasets. Table C.2 provides an overview of examples from these datasets.

### C.3.3 Concept Categorization Task.

The concept categorization task requires classifying nouns into a concept type derived from an ontology. For this task, we employ the AP Almuhareb and Poesio (2005), BLESS Baroni and Lenci (2011) and ESSL<sub>2b</sub> Marco Baroni and Lenci (2008) datasets. The AP dataset contains 402 nouns from 21 WordNet classes, e.g., nouns such as ‘ceremony’, ‘feast’, and ‘graduation’ belong to the class ‘Social Occasion’. The BLESS dataset, designed for the evaluation of distributional semantic models,

contains 200 distinct English concrete nouns as target concepts. These nouns are categorized into 17 broad classes.

### C.3.4 Evaluation Metrics and Pipeline.

The word similarity prediction effectiveness is measured with the help of Spearman’s rank correlation coefficient  $\rho$ . This measures the rank correlation (higher is better) between the list of word pairs sorted in decreasing order of inter-similarity values as predicted by a word embedding algorithm and the reference list of human judged word pairs. For the analogy and the concept categorization tasks, we report the accuracy in predicting the reference word and that of the class, respectively.

#### Parameters and Settings.

In our experiments, for all the methods, except ELMO, we set the number of dimensions to 200. To find optimal settings for each method (except ELMO), we use the MEN dataset as a development set for tuning the parameters of each method. Each method with the optimal parameter settings is then applied for the rest of the datasets and tasks.

Since we used a pre-trained model for ELMO, the number of dimensions corresponds to the size of the output layer of the network, the value of which in the default configuration of the Python implementation<sup>3</sup> is 1024.

The parameters of SGNS are window size ( $k$ ) and the number of negative samples (NS). For the baseline approach SGNS, we varied  $k$  from 5 to 40 in steps of 5 and found that the best results are obtained when  $k = 10$  and  $NS = 5$ . Similarly, for Glove we chose the optimal settings by varying  $k$  within the same range of  $[5, 40]$  and found that the optimal  $\rho$  for the MEN dataset is obtained for  $k = 20$ . We obtain the LDA results by setting the number of topics to 200 (so as to match with the dimensionality). As LDA hyper-parameters, we use settings as prescribed in Griffiths and Steyvers (2004), i.e.,  $\beta = 0.1$  and  $\alpha = 0.25$  ( $50/(\#topics = 200)$ ).

---

<sup>3</sup>[https://github.com/allenai/allennlp/blob/master/tutorials/how\\_to/elmo.md](https://github.com/allenai/allennlp/blob/master/tutorials/how_to/elmo.md)

Since we found that SGNS performed significantly better than Glove, we use SGNS vectors for the linear combination method, which we call **SGNS-LDA** from hereon. The parameter  $\lambda$  was varied within a range of  $[0.1, 0.9]$  in steps of 0.1 ( $\lambda = 0$  and  $\lambda = 1$  degenerate to that of LDA and SGNS respectively). We found that the best results are obtained for  $\lambda = 0.9$ .

For node2vec baseline approach of word-node embedding, we varied the parameters  $p$  and  $q$  (BFS and DFS parameters) within a range of  $[0.1, 5]$  and found that the best results on the MEN dataset are given for  $p = 1$  and  $q = 1$  Grover and Leskovec (2016). Another parameter in node2vec is the random walk length,  $l$ , for which the optimal value was found to be 80.

For word-node2vec, in addition to window size ( $k$ ) and number of negative samples ( $NS$ ), three more parameters are: i)  $\alpha$ , i.e., the importance of the presence of a term relative to its informativeness, ii)  $\beta$ , the prior assigned to sampling from the 1-adjacent neighborhood, and iii) the size of the context sampled from the neighborhood,  $l$  (this is analogous to the random walk length parameter of node2vec). Instead of separately optimizing the parameters common to SGNS, we directly use the optimal values of  $k = 10$  and  $NS = 5$  for word-node2vec. The optimal results of the additional parameters, tuned on the MEN dataset, are shown in Table C.3.

## C.4 Results

**Word Similarity Prediction.** Table C.3 shows the results obtained by the competing methods on the word similarity prediction task. It can be seen that Glove turns out to be relatively ineffective in modeling the semantic representations of words as compared to human judgments. SGNS performs significantly better and the settings trained on MEN dataset generalize well on the WSIM-353 dataset as well. LDA performs rather poorly indicating that only global co-occurrences can lead to noisy representations of words. FastText performs worse as compared to SGNS. It is worth mentioning that the performance of ELMO is disappointing on

this task of semantic similarity prediction, because of the most likely reason that it better learns vector representations of word in the presence of a context.

A linear combination of SGNS and LDA does not perform better than SGNS, which means that a simple way of combining the embedded representations obtained individually with local and non-local approaches does not work well.

The node2vec approach of embedding nodes of the word-nodes graph constructed relies on a random walk based construction of the context of a word node. This random walk based context construction is only able to improve the SGNS results slightly, indicating that random walks can introduce noise in the contexts of word-nodes.

The word-node based graph construction (incorporating local and non-local co-occurrences in a principled way) works particularly well in conjunction with the stratified sampling based approach of selecting context words from the  $\kappa$ -neighborhood. The optimal value of  $\alpha = 0.5$  suggests that document-level co-occurrences should be computed by assigning equal importance to term presence and informativeness. A value of  $\beta = 0.7$  confirms the hypothesis that more emphasis should be put on direct co-occurrences.

**Word Analogy and Concept Categorization.** Similar trends are observed in the word analogy and concept categorization tasks in Tables C.4 and C.5 respectively. Relatively higher improvements with word-node2vec are noted for the MSR analogy task (comprised of syntactic categories). Among the baseline approaches, both node2vec and SGNS-LDA work well on the concept categorization task. However, the performance improvements are inconsistent across datasets, e.g. SGNS-LDA performs well on ESSLI<sub>2b</sub> and poorly on AP. Our proposed method configured on the MEN dataset works consistently well across all datasets, which indicates that word-node2vec can generalize well for different tasks.

As a side observation, we note that ELMO performs well for the analogy and concept categorization tasks (yielding the best results in particular on the Google

Method	Spearman's $\rho$	
	MEN	WSIM
SGNS ( $k = 10, NS = 5$ )	0.7432	0.6977
Glove ( $k = 20$ )	0.7066	0.6706
FastText	0.7307	0.6518
ELMO	0.4225	0.4631
LDA	0.4933	0.4074
SGNS-LDA ( $\lambda = 0.9$ )	0.7367	0.6548
Node2vec ( $p = 0.5, q = 0.5, l = 40$ )	0.7440	0.6988
Word-node2vec ( $\alpha = 0.5, \beta = 0.7, l = 20$ )	<b>0.7491</b>	<b>0.7032</b>

Table C.3: Word similarity prediction results.

Method	Accuracy (P@1)		
	Google	MSR	SemEval'12
SGNS	0.5615	0.2777	0.1460
Glove	0.4841	0.2485	0.1419
FastText	0.4930	0.2607	<b>0.1592</b>
ELMO	<b>0.5986</b>	0.2789	0.1439
LDA	0.0578	0.0158	0.0596
SGNS-LDA	0.5491	0.2776	0.1413
Node2vec	0.5588	0.2785	0.1427
Word-node2vec	0.5627	<b>0.2890</b>	0.1464

Table C.4: Word analogy results.

analogy dataset). Although the results are not directly comparable because of differences in the dimensionality of the vectors and also in the collection of documents used in the pre-trained ELMO vectors (Billion word benchmark as against DBPedia in our case), it could possibly be reasoned that the additional contextual information of the ELMO vectors turns out to be useful for in the analogy task.

## C.5 Conclusions

We proposed a word embedding approach that leverages document-level non-local co-occurrences, in addition to the window-based local co-occurrences. We proposed a graph-based framework, in which words are represented as nodes and the edges between a pair of words reflect the degree of association between them. This association is a function of both the local and the document-level co-occurrences, which

Method	Accuracy		
	AP	BLESS	ESSLI <sub>2b</sub>
SGNS	0.6194	0.7500	0.7500
Glove	0.6343	0.7200	0.7250
FastText	0.6119	<b>0.7950</b>	0.7250
ELMO	0.6368	0.7350	0.7500
LDA	0.3383	0.3900	0.6500
SGNS-LDA	0.5796	0.7850	<b>0.7750</b>
Node2vec	0.6355	0.7500	0.7350
Word-node2vec	<b>0.6393</b>	<b>0.7950</b>	<b>0.7750</b>

Table C.5: Concept categorization results.

enables our approach to achieve ‘the best of both worlds’ in word embedding. Experiments show that our proposed method outperforms local approaches, namely word2vec, Glove and FastText, on a number of different tasks. Our approach also outperforms a naive black-box combination of embeddings obtained separately by local and document-level approaches. This proves the importance of addressing both these sources of information jointly in an embedding objective.



# Bibliography

- Almeida, F. and Xexéo, G. (2019). Word embeddings: A survey.
- Almuhareb, A. and Poesio, M. (2005). Concept learning and categorization from the web. In *Proc. of COGSCI*, pages 103–108.
- Bahl, L. R., Jelinek, F., and Mercer, R. L. (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):179–190.
- Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don’t count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proc. of ACL*, pages 238–247. Association for Computational Linguistics.
- Baroni, M. and Lenci, A. (2011). How we blessed distributional semantic evaluation. In *Proc. of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10.
- Ben Carterette, E. K., Hall, M., and Clough, P. (2014). Overview of the trec 2014 session track. In *Proc. of the Twenty Third Text REtrieval Conference (TREC 2014)*.
- Bengio, Y. and Senécal, J.-S. (2003). Quick training of probabilistic neural nets by importance sampling.
- Bernstein, M. S., Van Kleek, M., schraefel, m. c., and Karger, D. R. (2007). *Management of Personal Information Scraps*, page 2285–2290. Association for Computing Machinery, New York, NY, USA.

- Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.
- Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., and Vigna, S. (2008). The query-flow graph: Model and applications. In *Proc. of CIKM 2008*, pages 609–618.
- Bonchi, F., Perego, R., Silvestri, F., Vahabi, H., and Venturini, R. (2012). Efficient query recommendations in the long tail via center-piece subgraphs. In *Proc. of SIGIR 2012*, pages 345–354. ACM.
- Brondwine, E., Shtok, A., and Kurland, O. (2016). Utilizing focused relevance feedback. In *Proc. of SIGIR 2016*, SIGIR ’16, page 1061–1064.
- Bruni, E., Tran, N. K., and Baroni, M. (2014). Multimodal distributional semantics. *J. Artif. Int. Res.*, 49:1–47.
- Buscher, G., Dengel, A., and van Elst, L. (2008). Eye movements as implicit relevance feedback. In *CHI ’08 Extended Abstracts on Human Factors in Computing Systems*, page 2991–2996.
- Cai, Y., Dong, X. L., Halevy, A., Liu, J. M., and Madhavan, J. (2005). Personal information management with semex. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, page 921–923, New York, NY, USA. Association for Computing Machinery.
- Carterette, B., Kanoulas, E., Hall, M. M., and Clough, P. D. (2014). Overview of the TREC 2014 session track. In *Proc. of TREC ’14*.
- Chappell, T. and Geva, S. (2010). Overview of the INEX 2010 focused relevance feedback track. In *Comparative Evaluation of Focused Retrieval - 9th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2010*, pages 303–312.

- Chein, M. and Mugnier, M.-L. (2008). *A Graph-Based Approach to Knowledge Representation: Computational Foundations of Conceptual Graphs (Part. I)*.
- Chen, J., Guo, H., Wu, W., and Wang, W. (2009). imecho: An associative memory based desktop search system. In *Proceedings of CIKM*, pages 731–740.
- Chen, J., Guo, H., Wu, W., and Wang, W. (2011). Imecho: A context-aware desktop search system. In *Proc. of SIGIR 2011*, page 1269–1270.
- Chen, J., Guo, H., Wu, W., and Xie, C. (2010). Search your memory! - an associative memory based desktop search system. In *Proc. of SIGMOD*, page 1099–1102.
- Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning, ICML '08*, page 160–167, New York, NY, USA. Association for Computing Machinery.
- Cormack, G. V., Smucker, M. D., and Clarke, C. L. (2011). Efficient and effective spam filtering and re-ranking for large web datasets. *Inf. Retr.*, 14(5):441–465.
- Crouch, C. J., Crouch, D. B., and Nareddy, K. R. (1990). The automatic generation of extended queries. In *Proc. of SIGIR*, SIGIR '90, page 369–383.
- Dehghani, M., Zamani, H., Severyn, A., Kamps, J., and Croft, W. B. (2017). Neural ranking models with weak supervision. In *Proc. of SIGIR 2017*, pages 65–74.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL'19*, pages 4171–4186.
- Dumais, S., Cutrell, E., Cadiz, J. J., Jancke, G., Sarin, R., and Robbins, D. C. (2016). Stuff i've seen: A system for personal information retrieval and re-use. *SIGIR Forum*, pages 28–35.

- Dumais, S., Cutrell, E., Sarin, R., and Horvitz, E. (2004). Implicit queries (iq) for contextualized search. In *Proceedings of SIGIR*, pages 594–594.
- Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2015). Retrofitting word vectors to semantic lexicons. In *Proc. of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, Denver, Colorado. Association for Computational Linguistics.
- Feild, H. and Allan, J. (2013). Task-aware query recommendation. In *Proc. of SIGIR 2013*, pages 83–92.
- Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2014). Placing search in context: The concept revisited. In *Proc. of WWW 2014*, pages 406–414.
- Fitchett, S. and Cockburn, A. (2012). Accessrank: Predicting what users will do next. In *Proceedings of the SIGCHI*, pages 2239–2242.
- Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Ranzato, M., and Mikolov, T. (2013a). Devise: A deep visual-semantic embedding model. In *Proc. of NIPS’13*, pages 2121–2129.
- Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Ranzato, M. A., and Mikolov, T. (2013b). Devise: A deep visual-semantic embedding model. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 26, pages 2121–2129. Curran Associates, Inc.
- Futami, H., Inaguma, H., Ueno, S., Mimura, M., Sakai, S., and Kawahara, T. (2020). Distilling the knowledge of bert for sequence-to-sequence asr.
- Ganguly, D., Ganguly, M., Leveling, J., and Jones, G. J. F. (2013). Topicvis: a

- GUI for topic-based feedback and navigation. In *In Proc. of SIGIR'13*, pages 1103–1104.
- Ganguly, D., Leveling, J., Magdy, W., and Jones, G. J. (2011a). Patent query reduction using pseudo relevance feedback. In *Proc. of CIKM*, CIKM '11, page 1953–1956.
- Ganguly, D., Leveling, J., Magdy, W., and Jones, G. J. F. (2011b). Patent query reduction using pseudo relevance feedback. In Macdonald, C., Ounis, I., and Ruthven, I., editors, *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, pages 1953–1956. ACM.
- Goodman, J. (2001). Classes for fast maximum entropy training.
- Grbovic, M., Djuric, N., Radosavljevic, V., Silvestri, F., and Bhamidipati, N. (2015). Context- and content-aware embeddings for query rewriting in sponsored search. In *Proc. of SIGIR '15*, pages 383–392.
- Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235.
- Grover, A. and Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proc. of ACM SIGKDD'16*, pages 855–864.
- Han, F., Niu, D., Lai, K., Guo, W., He, Y., and Xu, Y. (2019). Inferring search queries from web documents via a graph-augmented sequence to attention network. In *The World Wide Web Conference*, page 2792–2798.
- Harman, D. (2002). Overview of the trec 2002 novelty track. In *Proc. of the Eleventh Text REtrieval Conference (TREC 2002), NIST Special Publication 500-251*, pages 46–55.
- Hassan Awadallah, A., White, R. W., Pantel, P., Dumais, S. T., and Wang, Y.-M. (2014). Supporting complex search tasks. In *Proc. of CIKM 2014*, pages 829–838.

- Hiemstra, D. (2000). *Using Language Models for Information Retrieval*. PhD thesis, Center of Telematics and Information Technology, AE Enschede.
- Hiemstra, D. and Kraaij, W. (2005). A language modeling approach for TREC. In Voorhees, E. M. and Harman, D., editors, *TREC: Experiment and Evaluation in Information Retrieval*. MIT press.
- Hinbarji, Z., Hinbarji, M., Albatal, R., and Gurrin, C. (2016). Personal information manager to capture and re-access what we see on computers. In *Proceedings of the First Workshop on Lifelogging Tools and Applications*, LTA '16, page 13–17. Association for Computing Machinery.
- Hu, Y. and Janowicz, K. (2012). Improving personal information management by integrating activities in the physical world with the semantic desktop. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, page 578–581, New York, NY, USA. Association for Computing Machinery.
- Järvelin, K. (2010). Test collections and evaluation metrics based on graded relevance. In Majumder, P., Mitra, M., Bhattacharyya, P., Subramaniam, L. V., Contractor, D., and Rosso, P., editors, *Multilingual Information Access in South Asian Languages - Second International Workshop, FIRE 2010, Gandhinagar, India, February 19-21, 2010 and Third International Workshop, FIRE 2011, Bombay, India, December 2-4, 2011, Revised Selected Papers*, Lecture Notes in Computer Science, pages 280–294. Springer.
- Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.
- Jelinek, F. and Mercer, R. L. (1980). Interpolated estimation of Markov source parameters from sparse data. In Gelsema, E. S. and Kanal, L. N., editors, *Proceedings, Workshop on Pattern Recognition in Practice*, pages 381–397. North Holland, Amsterdam.

- Jones, R. and Klinkner, K. L. (2008). Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *Proc. of CIKM '08*, pages 699–708.
- Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016). Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.
- Jurgens, D. A., Turney, P. D., Mohammad, S. M., and Holyoak, K. J. (2012). Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proc. of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proc. of the Main Conference and the Shared Task, and Volume 2: Proc. of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 356–364.
- Kanoulas, E., Carterette, B., Clough, P. D., and Sanderson, M. (2011). Evaluating multi-query sessions. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 1053–1062, New York, NY, USA. Association for Computing Machinery.
- Karimzadehgan, M. and Zhai, C. (2010). Estimation of statistical translation models based on mutual information for ad hoc information retrieval. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, page 323–330, New York, NY, USA. Association for Computing Machinery.
- Karimzadehgan, M. and Zhai, C. (2012). Axiomatic analysis of translation language model for information retrieval. In *Proceedings of the 34th European Conference on Advances in Information Retrieval*, ECIR'12, page 268–280, Berlin, Heidelberg. Springer-Verlag.
- Kelly, D. and Azzopardi, L. (2015a). How many results per page?: A study of SERP size, search behavior and user experience. In Baeza-Yates, R., Lalmas, M., Moffat, A., and Ribeiro-Neto, B. A., editors, *Proc. of the 38th International*

- ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 183–192. ACM.
- Kelly, D. and Azzopardi, L. (2015b). How many results per page? a study of serp size, search behavior and user experience. In *Proc. of SIGIR 2015*, page 183–192.
- Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.
- Khattab, O. and Zaharia, M. (2020). Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In Huang, J., Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J., and Liu, Y., editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48. ACM.
- Kim, J., Bakalov, A., Smith, D. A., and Croft, W. B. (2010). Building a semantic representation for personal information. In *Proceedings of CIKM*, pages 1741–1744.
- Kong, W., Li, R., Luo, J., Zhang, A., Chang, Y., and Allan, J. (2015). Predicting search intent based on pre-search context. In *Proc. of SIGIR 2015*, pages 503–512.
- Koskela, M., Luukkonen, P., Ruotsalo, T., Sjöberg, M., and Floréen, P. (2018). Proactive information retrieval by capturing search intent from primary task context. *ACM Trans. Interact. Intell. Syst.*, pages 20:1–20:25.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105.
- Lavrenko, V. and Croft, W. B. (2001). Relevance based language models. In *Proc. of SIGIR 2001*, pages 120–127.



- Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proc. of ICML 2014*, pages II–1188–II–1196.
- Lee, C.-J. and Croft, W. B. (2012). Generating queries from user-selected text. In *Proc. of the 4th Information Interaction in Context Symposium*, page 100–109.
- Levine, N., Roitman, H., and Cohen, D. (2017). An extended relevance model for session search. In *Proc. of SIGIR 2017*, pages 865–868.
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.
- Li, L., Deng, H., Dong, A., Chang, Y., Baeza-Yates, R., and Zha, H. (2017). Exploring query auto-completion and click logs for contextual-aware web search and query suggestion. WWW ’17, page 539–548, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.
- Li, R., Kao, B., Bi, B., Cheng, R., and Lo, E. (2012). Dqr: A probabilistic approach to diversified query recommendation. In *Proc. of CIKM 2012*, pages 16–25.
- Li, S., Zhu, J., and Miao, C. (2015). A generative word embedding model and its low rank positive semidefinite solution. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1599–1609, Lisbon, Portugal. Association for Computational Linguistics.
- Liu, J., Liu, C., and Belkin, N. J. (2016). Predicting information searchers’ topic knowledge at different search stages. *JASIST*, pages 2652–2666.
- Lucchese, C., Orlando, S., Perego, R., Silvestri, F., and Tolomei, G. (2013). Discovering tasks from search engine query logs. *ACM Trans. Inf. Syst.*, 31(3):14:1–14:43.
- Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, USA.

- Marco Baroni, S. E. and Lenci, A. (2008). Esslli workshop on distributional lexical semantics. *ESSLLI Workshop on Distributional Lexical Semantics*, 101:1–70.
- Maxwell, D. and Azzopardi, L. (2016a). Agents, simulated users and humans: An analysis of performance and behaviour. In *Proc. of CIKM 2016*, page 731–740.
- Maxwell, D. and Azzopardi, L. (2016b). Simulating interactive information retrieval: Simiir: A framework for the simulation of interaction. In *Proc. of SIGIR 2016*, page 1141–1144.
- Mehrotra, R., Bhattacharya, P., and Yilmaz, E. (2016). Deconstructing complex search tasks: a bayesian nonparametric approach for extracting sub-tasks. In *Proc. of NAACL HLT '16*, pages 599–605.
- Mehrotra, R. and Yilmaz, E. (2017a). Extracting hierarchies of search tasks and subtasks via a bayesian nonparametric approach. In *Proc. of SIGIR'17*, pages 285–294.
- Mehrotra, R. and Yilmaz, E. (2017b). Task embeddings: Learning query embeddings using task context. In *Proc. of CIKM 2017*, pages 2199–2202.
- Mikolov, T., Kopecky, J., Burget, L., Glembek, O., and ?Cernocky, J. (2009). Neural network based language models for highly inflective languages. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4725–4728.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013a). Distributed representations of words and phrases and their compositionality. In *Proc. NIPS '13*, pages 3111–3119.
- Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Proc. of NAACL 2013*, pages 746–751.
- Mitra, B., Shokouhi, M., Radlinski, F., and Hofmann, K. (2014). On user interactions with query auto-completion. In *Proceedings of the 37th International ACM*

- SIGIR Conference on Research Development in Information Retrieval*, pages 1055–1058.
- Mnih, A. and Hinton, G. (2007). Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 641–648, New York, NY, USA. Association for Computing Machinery.
- Mnih, A. and Hinton, G. E. (2009). A scalable hierarchical distributed language model. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 21, pages 1081–1088. Curran Associates, Inc.
- Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *AISTATS'05*, pages 246–252.
- Muntean, C. I., Nardini, F. M., Silvestri, F., and Sydow, M. (2013). Learning to shorten query sessions. In *Proc. of WWW 2013*, pages 131–132.
- Nogueira, R. and Cho, K. (2020). Passage re-ranking with bert.
- Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proc. of EMNLP 2014*, pages 1532–1543.
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proc. of NAACL'18*.
- Pise, P. D. and Uke, N. J. (2016). Efficient security framework for sensitive data sharing and privacy preserving on big-data and cloud platforms. In *Proceedings of the International Conference on Internet of Things and Cloud Computing*, New York, NY, USA. Association for Computing Machinery.
- Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proc. of SIGIR 1998*, pages 275–281.

- Puolamäki, K., Salojärvi, J., Savia, E., Simola, J., and Kaski, S. (2005). Combining eye movements and collaborative filtering for proactive information retrieval. In *Proceedings of SIGIR 2005*, pages 146–153.
- Qiu, Y. and Frei, H.-P. (1993). Concept based query expansion. In *Proc. of ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93, page 160–169, New York, NY, USA. Association for Computing Machinery.
- Qvarfordt, P., Golovchinsky, G., Dunnigan, T., and Agapie, E. (2013). Looking ahead: Query preview in exploratory search. In *Proc. of SIGIR 2013*, pages 243–252.
- Rhodes, B. (2000). The wearable remembrance agent: A system for augmented memory. *Personal Technologies*, 1.
- Rhodes, B. J. and Maes, P. (2000). Just-in-time information retrieval agents. *IBM Syst. J.*, 39(3–4):685–704.
- Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. page 333–389.
- Robertson, S. E. (1977). The Probability Ranking Principle in IR. *Journal of Documentation*, 33(4):294–304.
- Robertson, S. E. (1990). On term selection for query expansion. *J. Documentation*, 46(4):359–364.
- Robertson, S. E., Kanoulas, E., and Yilmaz, E. (2010). Extending average precision to graded relevance judgments. In *SIGIR*, pages 603–610. ACM.
- Robertson, S. E., Walker, S., Jones, S., and Hancock-Beaulieu, M. (1994). Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC 1994)*. NIST.

- Rothe, S., Narayan, S., and Severyn, A. (2020). Leveraging pre-trained checkpoints for sequence generation tasks.
- Roy, D., Ganguly, D., Mitra, M., and Jones, G. J. (2016). Word vector compositionality based relevance feedback using kernel density estimation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM '16*, page 1281–1290, New York, NY, USA. Association for Computing Machinery.
- Salakhutdinov, R. and Mnih, A. (2008). Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proc. of ICML 2008*, pages 880–887.
- Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523.
- Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.
- Santos, R. L., Macdonald, C., and Ounis, I. (2010). Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th International Conference on World Wide Web, WWW '10*, page 881–890, New York, NY, USA. Association for Computing Machinery.
- Sen, P., Ganguly, D., and Jones, G. (2018a). Procrastination is the thief of time: Evaluating the effectiveness of proactive search systems. In *Proc. of SIGIR 2018*, pages 1157–1160.
- Sen, P., Ganguly, D., and Jones, G. (2019). Word-Node2Vec: Improving word embedding with document-level non-local word co-occurrences. In *Proc. of NAACL*, pages 1041–1051, Minneapolis, Minnesota. Association for Computational Linguistics.
- Sen, P., Ganguly, D., and Jones, G. J. (2018b). Tempo-lexical context driven word

- embedding for cross-session search task extraction. In *Proc of NAACL-HLT 2018*, pages 283–292.
- Shaw, J. A., Fox, E. A., Shaw, J. A., and Fox, E. A. (1994). Combination of multiple searches. In *Proc. of TREC-2 1994*, pages 243–252.
- Shokouhi, M. and Guo, Q. (2015). From queries to cards: Re-ranking proactive card recommendations based on reactive search history. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 695–704, New York, NY, USA. Association for Computing Machinery.
- Singhal, A. (1997). *Term Weighting Revisited*. PhD thesis, Cornell University, Department of computer science.
- smart (2019). SMART stopword list. <https://www.lextek.com/manuals/onix/stopwords2.html>,.
- Smyth, B., Balfe, E., Briggs, P., Coyle, M., and Freyne, J. (2003). I-spy - anonymous, community-based personalization by collaborative meta-search.
- Smyth, B., Briggs, P., Coyle, M., and O’Mahony, M. P. (2009). Google shared. A case-study in social search. In Houben, G., McCalla, G. I., Pianesi, F., and Zancanaro, M., editors, *User Modeling, Adaptation, and Personalization, 17th International Conference, UMAP 2009, formerly UM and AH, Trento, Italy, June 22-26, 2009. Proceedings*, volume 5535 of *Lecture Notes in Computer Science*, pages 283–294. Springer.
- Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. (2011). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK. Association for Computational Linguistics.

- Somlo, G. L. and Howe, A. E. (2003). Using web helper agent profiles in query generation. In *Proc. of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, page 812–818.
- Song, Y. and Guo, Q. (2016). Query-less: Predicting task repetition for nextgen proactive search and recommendation engines. In *Proceedings of the 25th International Conference on World Wide Web, WWW '16*, page 543–553, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.
- Sordoni, A., Bengio, Y., Vahabi, H., Lioma, C., Grue Simonsen, J., and Nie, J.-Y. (2015). A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proc. of CIKM 2015*, pages 553–562.
- Sparck Jones, K. (1973). Index term weighting. *Journal of Documentation*, 9(11):619–633.
- Sparck-Jones, K., Walker, S., and Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management*, 36(6):779–840.
- Takaki, T., Fujii, A., and Ishikawa, T. (2004). Associative document retrieval by query subtopic analysis and its application to invalidity patent search. In *Proc. of CIKM'04*, pages 399–405.
- Tan, B., Shen, X., and Zhai, C. (2006). Mining long-term search history to improve search accuracy. In *Proc. of SIGKDD 2006*, pages 718–723.
- Tran, T. A., Schwarz, S., Niederée, C., Maus, H., and Kanhabua, N. (2016). The forgotten needle in my collections: Task-aware ranking of documents in semantic information space. In *Proceedings of CHIIR*, pages 13–22.
- Tripathy, A. and Pradhan, M. (2012). A novel framework for preserving privacy of data using correlation analysis. In *Proceedings of the International Conference on*

- Advances in Computing, Communications and Informatics*, page 650–655, New York, NY, USA. Association for Computing Machinery.
- Van Gysel, C., Kanoulas, E., and de Rijke, M. (2016). Lexical query modeling in session search. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, page 69–72, New York, NY, USA. Association for Computing Machinery.
- Van Gysel, C., Mitra, B., Venanzi, M., Rosemarin, R., Kukla, G., Grudzien, P., and Cancedda, N. (2017). Reply with: Proactive recommendation of email attachments. In *Proc. of CIKM 2017*, page 327–336.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.
- Verma, M. and Yilmaz, E. (2014). Entity oriented task extraction from query logs. In *Proc. of CIKM’14*, pages 1975–1978.
- Vieira, M. R., Razente, H. L., Barioni, M. C. N., Hadjieleftheriou, M., Srivastava, D., Traina, C., and Tsotras, V. J. (2011). Divddb: A system for diversifying query results. *Proc. VLDB Endow.*, 4(12):1395–1398.
- Voorhees, E. M. (2003). Overview of the TREC 2003 robust retrieval track. In Voorhees, E. M. and Buckland, L. P., editors, *Proc. of The Twelfth Text REtrieval Conference, TREC 2003, Gaithersburg, Maryland, USA, November 18-21, 2003*, NIST Special Publication, pages 69–77. National Institute of Standards and Technology (NIST).
- Vuong, T., Jacucci, G., and Ruotsalo, T. (2017). Proactive information retrieval via screen surveillance. In *Proceedings of SIGIR 2017*, pages 1313–1316.
- Wang, H., Song, Y., Chang, M.-W., He, X., White, R. W., and Chu, W. (2013).



- Learning to extract cross-session search tasks. In *Proc. of WWW '13*, pages 1353–1364.
- Wei, X. and Croft, W. B. (2006). Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 178–185, New York, NY, USA. Association for Computing Machinery.
- White, R. W. and Morris, D. (2007). Investigating the querying and browsing behavior of advanced search engine users. In *Proc. of SIGIR*, page 255–262.
- Xue, X. and Croft, W. B. (2009). Automatic query generation for patent search. In *Proc. of CIKM*, page 2037–2040.
- Yang, L., Guo, Q., Song, Y., Meng, S., Shokouhi, M., McDonald, K., and Croft, W. B. (2016). Modelling user interest for zero-query ranking. In *European Conference on Information Retrieval (ECIR 2016)*.
- Yang, Y., Bansal, N., Dakka, W., Ipeirotis, P., Koudas, N., and Papadias, D. (2009). Query by document. In *Proc. of WSDM 2009*, page 34–43.
- Yilmaz, Z. A., Wang, S., Yang, W., Zhang, H., and Lin, J. (2019). Applying BERT to document retrieval with birch. pages 19–24.
- Yoshua Bengio, Réjean Ducharme, P. V. and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.
- Yu, X., Ma, H., Hsu, B.-J. P., and Han, J. (2014). On building entity recommender systems using user click log and freebase knowledge. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining, WSDM '14*, page 263–272, New York, NY, USA. Association for Computing Machinery.
- Zamani, H. and Croft, W. B. (2017). Relevance-based word embedding. In *Proc. of SIGIR'17*, pages 505–514.

- Zhang, S., Guan, D., and Yang, H. (2013). Query change as relevance feedback in session search. In *Proc. of SIGIR 2013*, pages 821–824.
- Zheng, G. and Callan, J. (2015). Learning to reweight terms with distributed representations. In *Proc. of SIGIR '15*, pages 575–584.